A REMOTE ONLINE EXPERIMENT AT THE UNIVERSITY OF QUEENSLAND

By

Joel Carpenter

THE UNIVERSITY OF QUEENSLAND

The School of Information Technology and

Electrical Engineering

Submitted for the degree of

Bachelor of Engineering (Computer Systems)

OCTOBER 26th 2005

Head of School
School of Engineering
University of Queensland
St Lucia, Q 4072

Dear Professor Bailes,

In accordance with the requirements of the degree of Bachelor of Engineering I present the following thesis entitled

"A remote laboratory at the University of Queensland"

This work was performed under the supervision of Dr. Mark Schulz. I declare that the work submitted in this thesis is my own, except as acknowledged in the text and footnotes, and has not been previously submitted for a degree at the University of Queensland or any other institution.

Yours sincerely,

Joel Carpenter

To my family…

Acknowledgements

Firstly, I would like to thank Mark Schulz for giving me the opportunity to do this topic and his assistance with writing this document.

To Judson Harward, Charuleka Varadharajan, Imad Jabbour, Kirky Ringer DeLong and the rest of the MIT iLab team your assistance was enthusiastic, prompt, helpful and most appreciated. It gave me real peace of mind to know that when problems struck, I had a team at MIT to help me out.

Next I would like to thank Geir Hovland and Shane Goodwin for their assistance regarding the inverted pendulum experiment.

Caught somewhere in the cross-fire was Benn Cizauskas who's assistance with my the ITEE network infrastructure requirements was greatly appreciated and won't be forgotten.

And of course my family, especially my mother who is currently mowing the lawn I was supposed to do.

Abstract

This thesis describes the implementation of UQ's and Australia's first iLab experiment. An iLab is a remotely accessible laboratory that is part of a larger multi-million dollar joint MIT-Microsoft project called iCampus.

The iLab architecture was developed by MIT to support a scalable community of online experiments available for shared use by multiple institutions worldwide. It endeavours to create a shift in both the way labs are used and also how they are budgeted. Web-based access to labs allows students and staff to use labs at their leisure from wherever they may be, and with the institutions spanning time-zones, labs need no longer sit idle whilst the locals sleep. Not only can established universities benefit both academically and financially, but institutions in developing nations can be given the opportunity to work with equipment that they would never be able to otherwise.

The first iLab to go online at UQ will be the pole balancer [inverted pendulum] experiment currently undertaken in METR4202. It will see an iLab interfaced with the MATLAB xPC prototyping toolkit which will provide an excellent base for the development of many other varied iLabs. As this is the first iLab to be created at UQ the infrastructure will also need to be established and some general investigation and understanding of the iLab architecture will need to be addressed.

UQ has been designated by MIT as the Australasian hub for the iCampus project and the implementation of this first iLab is the initial step in UQ's responsibility to expand the project throughout our region

List of Figures

Figure 2.1 – Java TCP/IP Remote laboratory 3

Figure 2.2 - LabView remote panels 4

Figure 2.3 – Robot arm manipulation using LabView and telelabs 5

Figure 2.4 - iLab infrastructure 6

Figure 2.5 - Software layers of iLab with Java Proxy Service Broker Interface 11

Figure 3.1 – S-18 TLT from Signet Electronics 14

Figure 3.2 – Inverted pendulum 15

Figure 4.1 – Graphical redesign of Service Broker web interface 22

Figure 4.2 – Webcam webpage 31

Figure 4.3 – Example experiment result XML object 35

Figure 4.4 – Production of composite background image 40

Figure 4.5 – Padding of datasets to maintain speed and ease of use 41

Figure 4.6 – UQ network infrastructure 43

Figure 5.1 – Time spend logged onto Service Broker by hour 47

Figure 5.2 – Experiments conducted per hour of the day 47

List of Tables

Table 1: Installation variations investigated in location of Service Broker fault 21

Table 2 – Comparison of JFreechart and Chart2D 26

Table of Contents

1 – Introduction

1.1 Problem description and motivation 1

1.2 Project goals 1

1.3 Chapter Overview 1

2 – Background

2.1 – Java TCP/IP 3

2.2 – LabView 4

2.3 – Telelab Project 5

2.4 – iLab Project [MIT] 5

2.4.1 – Infrastructure 7

2.4.2 – Experiment Types 7

2.4.3 – Software 8

2.4.3.1 – iLab API 9

2.4.3.2 – Java Proxy 11

3 – Design Specification

3.1 – Reasons for the choice of iLabs 13

3.2 – Experiment choices 13

3.2.1 – Slotted Line 13

3.2.2 – AC Machine 14

3.2.3 – Inverted Pendulum 14

3.3 – iLab specifications 16

3.3.1 – Service Broker 16

3.3.2 – Lab Server 17

3.3.3 – Lab Client 17

4 – Prototype Design

4.1 – Service Broker 19

4.1.1 – Creation 21

4.1.2 – Graphical Redesign 21

4.2 – Prototypes 23

4.2.1 – Overview 23

4.2.2 – Time-of-Day (Basic Functionality) 23

4.2.3 – Capital City (RS-232) 24

4.2.4 – Sine Wave Server 25

4.2.5 – Inverted Pendulum (Final prototype) 27

4.2.5.1 – Lab Server Overview 27

4.2.5.2 – Lab Client Overview 27

4.2.5.3 – xPC Interface [Matlab] 28

4.2.5.3.1 – xPC Web Interface 28

4.2.5.3.2 – xPC API 28

4.2.5.4 – email Notification 29

4.2.5.5 – Webcam 30

4.2.5.6 – Simulink File Upload 32

4.2.5.7 – Experiment Validation 34

4.2.5.8 – Experiment Result 35

4.2.5.9 – GUI Overview 36

4.2.5.10 – Animation 36

4.2.5.10.1 – Timer 36

4.2.5.10.2 – Chart 37

4.2.5.10.3 – State Machine 38

4.2.5.10.4 – Pendulum 38

4.2.5.11 – Comparison of Results 40

4.2.5.12 – Feedback Form 42

4.2.5.13 – Integration into ITS/ITEE domain 42

5 – Product Evaluation 45

6 – Future Developments

6.1 – Integration into the ITEE or ITS domain 51

6.2 – iLab 6.0 Functionality 51

6.3 – Magnetic dampening of pendulum 51

6.4 – Lab Server Sub-Broker 51

6.5 – SMS Notification 52

6.6 – Interactive Experimentation 53

6.6.1 – Real-time Graphing 53

6.6.2 – Real-time Control 53

6.7 – HTML Client 53

6.8 – Expansion of Experiment Diversity 54

7 - Conclusions 55

8 – References 57

APPENDICES 59

1 – Introduction

1.1 – Problem description and motivation

Laboratory access is an essential part of any technical education. However labs are expensive to maintain and often access is limited. The proliferation of the internet, and in particular broadband, has diminished the significance of location and brought many distance objects to our fingertips. This ability can be extended to the lab, to bring access to equipment where there was none prior and compliment existing practical education.

1.2 – Project goals

The goal is to more effectively utilise this often expensive equipment and increase its accessibility to students. Whilst online access is not intended to replace physical access to equipment the experience conveyed through remote use should remain as close as possible to being physically present in the lab with all the same challenges and opportunities to learn. Additionally this new method of interfacing with the lab equipment should exploit any opportunities it has to compliment or extend the options currently available in the lab. Furthermore, as this is envisioned to be the first in a series of experiments to go online at UQ its development should be mindful of the potential to pass on what is learnt and implemented here to future online laboratories.

At the completion of this project an experiment will be made available for online use by students of a course running UQ as part of their practical education.

1.3 – Chapter Overview

The remainder of this thesis is divided into six chapters

Chapter 2 – Background

Provides information regarding different approaches and initiatives regarding online laboratory access with special attention to the MIT iLab project.

Chapter 3 – Design Specification

Outlines the rationale behind the choice of iLabs and the choice of experiment hardware. It further details the requirement of each of the three-tiers of the iLab framework

Chapter 4 – Prototype Design

Explains the creation and investigation of each of the three-tiers by the creation of successive prototypes implementing different functionalities, culminating in the final Inverted pendulum iLab.

Chapter 5 – Product Evaluation

Gauges the outcome of the final product with special attention to its effect on students and staff within METR4202.

Chapter 6 – Future Developments

Discusses potential improvements to the iLab created in this project and outlines more generally potential future research under the iLab project at UQ.

Chapter 7 – Conclusion

A brief description of the outcomes of this project

2 – Background

Universities have been putting experiments online since the dawn of the internet and a great many having been made available for remote usage in that time. However the complexity, relevance and scope of the experiments vary greatly. Many are proof of concept style operations focused on researching the steps and feasibility of online experiment conduction. Others expand existing software built to operate the equipment such as LabView to extend its scope of usage out of the lab and onto the internet. What follows is a brief look at some of the remote laboratory strategies currently in use around the world.

2.1 - Java TCP/IP

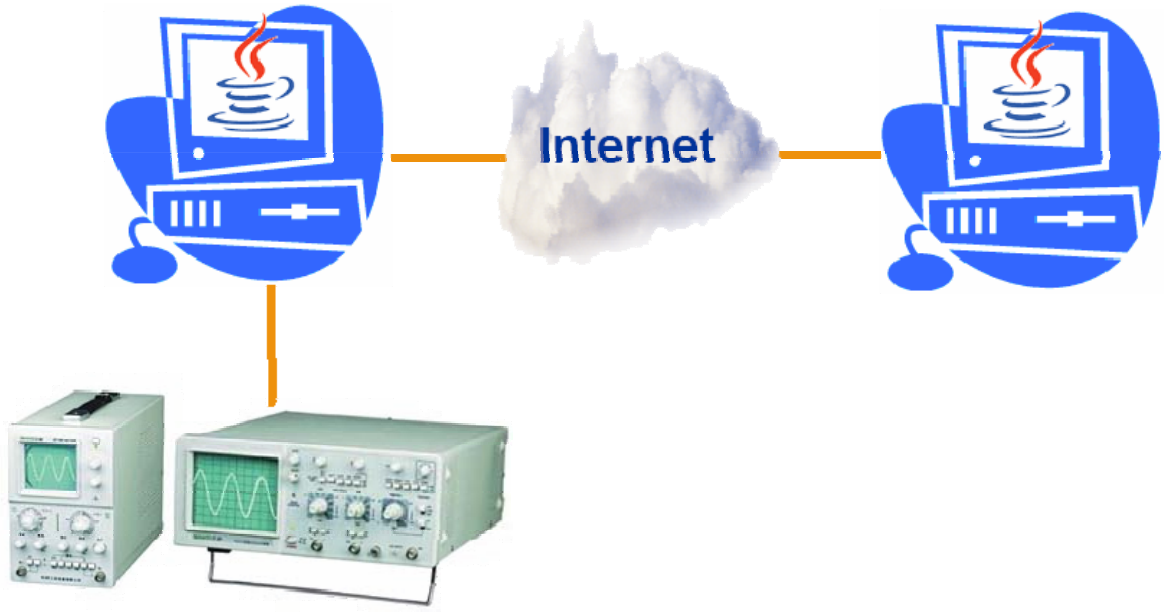


Figure 2.1 – Java TCP/IP Remote laboratory

This is by far the most common approach for one off online laboratories. A Java applet is run on the client computer which directly communicates using TCP/IP with the computer that interfaces with the equipment. This approach makes use of the power of Java and its integrated use with the web browser. This approach is functional; however it lacks the scalability of alternate solutions. Many of the earliest online laboratories used this approach.

2.2 - LabView [2]

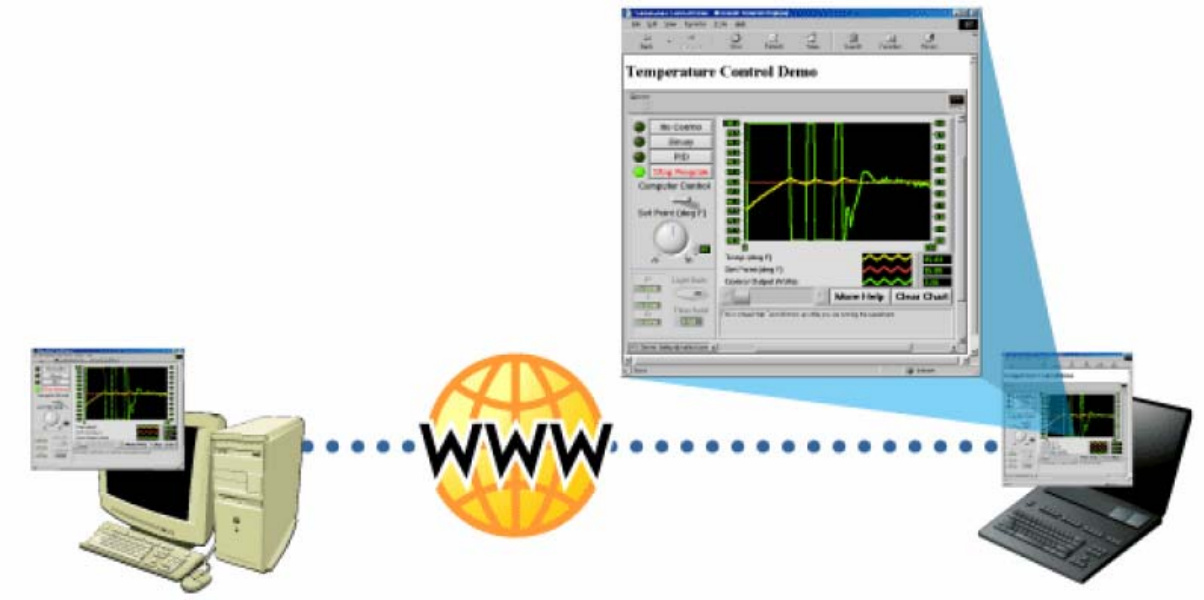


Figure 2.2[1] *- LabView remote panels*

LabView stands for *Laboratory Virtual Instrument Engineering Workbench*. It is a graphical programming language developed for the control of scientific and engineering equipment. Users develop applications typically for data acquisition, test and measurement or automation and control using a dataflow and block diagram style of specification. Interconnected icons represent functions, structures and instruments. With LabView users can develop a *front-panel*; a GUI interface used to interact with equipment at runtime. This front-panel can also be served to the internet as a *Remote-panel*. LabView is not designed as a scalable architecture for the creation of a large number of online labs, rather one client logs onto one server which controls one experiment. This pre-packaged, straight-from-the-box functionality has made it a popular approach for remote instrumentation. Online users must install the LabView runtime environment (around 35Mb) to operate the equipment. Such leading universities as Stanford[3] and the Swiss Federal Institute of Technology have utilised this product to put experiments online.

2.3 - Telelab Project (University of Western Australia)

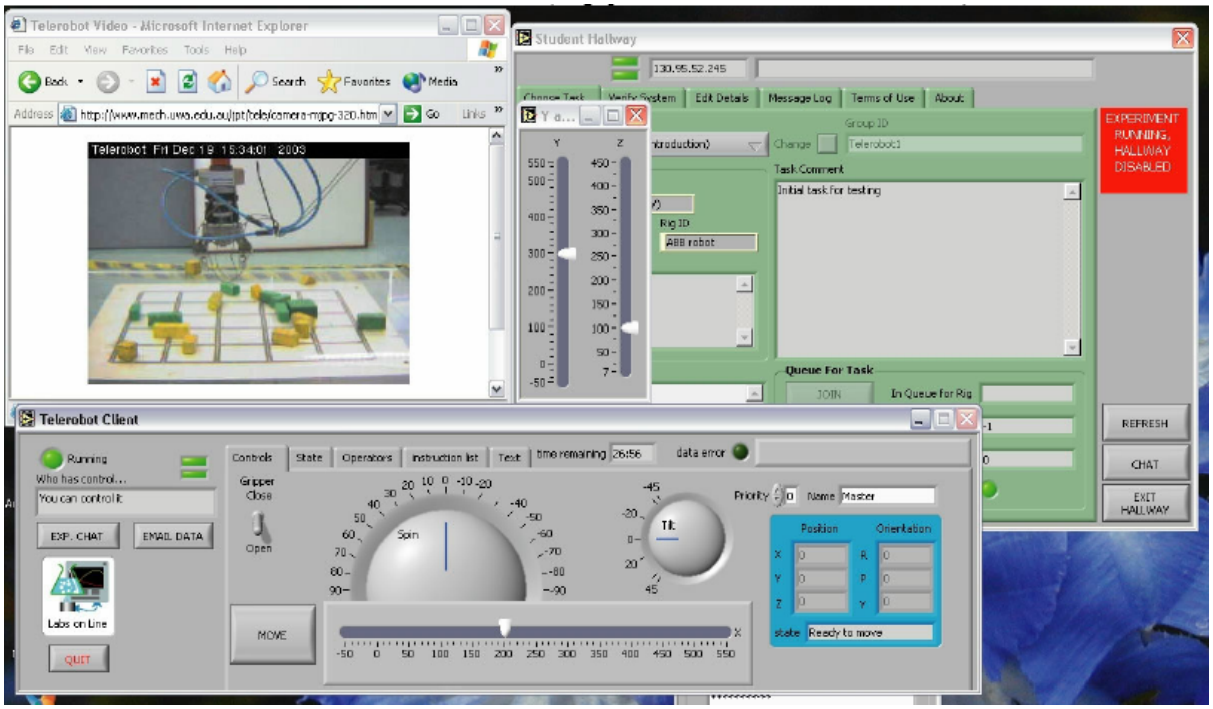


Figure 2.3 [4]– Robot arm manipulation using LabView and telelabs

Telelabs is a broad scope research initiative focused on using the internet to enhance student's learning. This project produced one of the earliest remote experiments, placing a mechanical arm online in 1994. This remote arm was controlled using CGI scripts initially but was upgraded in 2001 with the use of Java. Now the arm is controlled using LabView however the telelab project is investigating more scalable solutions to online experimentation that involve centralised servers capable of handling authentication, administration of multiple experiments and the archiving of experiment results. The telelabs has several other experiments online such as a thermodynamics practical that interfaces with sensors on a domestic iron and a torsional vibration experiment. Its philosophy is similar to the iLab project at MIT.

2.4 - iLab Project (Massachusetts Institute of Technology)

In 2000 Microsoft entered into a research initiative with MIT called iCampus. Its goal is to enhanced traditional modes of learning through the use of the latest information technology. A section of this project designed for teaching in engineering and science is iLab. The iLab project aims to create a worldwide community of educational institutions sharing resources through online access to laboratory equipment. In addition to this is also endeavours to

complement traditional labs in bringing availability were there was none prior, be it as a student using the equipment from home, or under-privileged countries gaining access to equipment at top universities.

2.4.1 - Infrastructure

The iLab architecture is designed as a general base for the development of online experiments with TCP/IP communication, authentication, administration and archiving of results already implemented. The developer builds on this existing backbone to create the Lab Client and the Lab Servers at either end of the data-path.

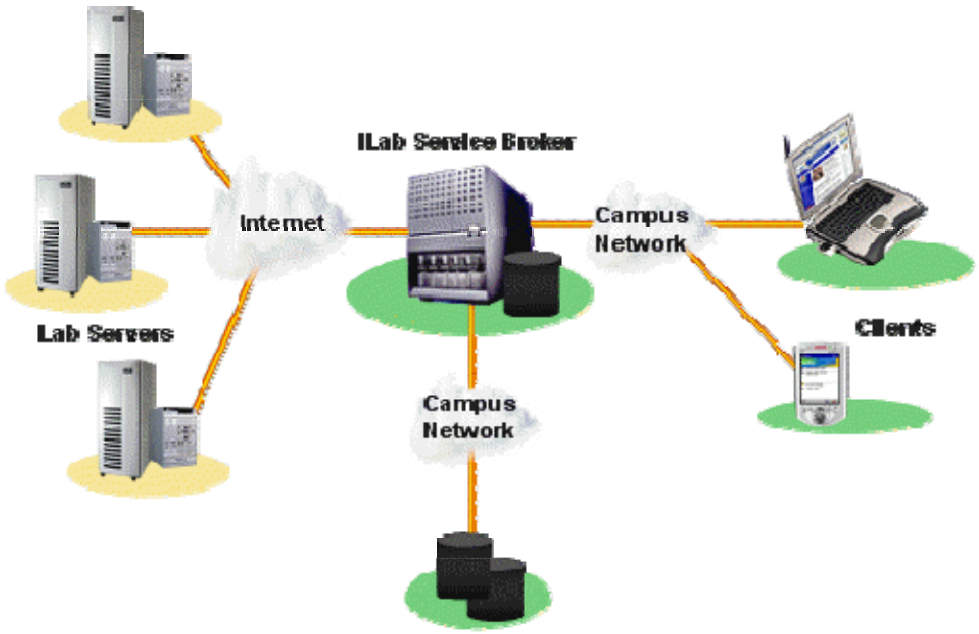


Figure 2.4 - *iLab infrastructure* [5]

iLab's design separates online labs into three distinct modules connected by a Web service architecture.

- The Lab Server is operated by the lab's owner and deals with the actual operation of the lab hardware.
- The Lab Client runs on the end user's computer, and provides the interface to the operation of the lab.
- The Service Broker mediates exchanges between the Lab Client and the Lab Server and provides storage and administrative services that are generic and can be shared by multiple labs within a single university.

The iLab project is funded by Microsoft Research. Consequently the infrastructure is Windows based, although the labs can be used by anyone with a Java enabled web browser.

Service Broker

This system consists primarily of;

- Windows Server 2003 Enterprise Edition
- SQL Server 2000 SP4 (latest service pack).
- Visual Studio .NET 2003 (for creation and debugging of content)
- IIS 5.0+

This is the backbone of the iLab framework. The user and the Lab Client communicate exclusively through the Service Broker which in turn passes messages through to the Lab Server. The Lab Client and Lab Server never exchange communications directly.

Lab Server

This system consists primarily of;

- Windows 2000, XP or Server (with .NET framework)
- IIS 5.0+
- Visual Studio .NET 2003 (for debugging)

This is the machine that interfaces with the lab equipment and responds to requests and control from the Service Broker.

Lab Client

Any machine running with a Java enabled web browser. The client applet is served up from the Service Broker. The latest version [6.0] of the iLab architecture also provides support for HTML clients. This can be any machine that can see the Service Broker either through a local intranet or the internet.

2.4.2 - Infrastructure

There are three general types of experiments a laboratory can undertake[1];

Batched – The entire specification of the experiment is determined before execution begins. The user need not remain online while the experiment executes.

Interactive – The most complicated type. The experiment is actively controlled by the user. It is possible that some initial conditions are set, but the user can continue to alter parameters whilst the experiment is running.

Sensor – A sensor sends streaming data back to the user. An online thermometer is a simple example. The flagpole at MIT (flagpole.mit.edu) is a more complicated sensor based experiment.

The iLab development team is addressing these different experiment types in successive releases of its iLab framework. Batched experiments are supported by the 5.0 framework which was under development from 2003 to 2005. Interactive experiments were first supported in the 6.0 framework in development from 2004 until 2006. Finally sensor experiment supported will be introduced and has just commenced development until a planned 2007.

2.4.3 – Software [8]

The iLab framework is deliberately general in its implementation as it requires the ability to interface with a wide range of experiments. Whilst a solid framework is in place through the centre of the dataflow the ends are dangling and it is up to the developer to tie them down. In the future there are hopes that the corporate sector will begin creating plug-n-play hardware ready built for the iLab environment. National Instruments already produce some web accessible lab monitoring software (LabView) but is not built into the iLab framework[1].

The software is built using Visual Studio .NET 2003 and the SDK including full source code and solution files are available at the iLabs Architecture downloads page[3]. The entire architecture is built around web services, SOAP (XML) and all client-Service Broker communication takes places over standard HTTP protocol. This ensures maximum interoperability between different networks from different institutions each with their own infrastructure, operating systems and security and firewall arrangements. The use of web services also allows for the integration of existing control software that comes with some lab equipment.

2.4.3.1 – API [11,12]

The Lab Client has a number of methods it can invoke on the Service Broker, and most of these are simply pass-through calls which are in turn simply passed on to the Lab Server.

Pass-through methods

- *Validate*

  Checks whether an experiment specification would be accepted if submitted for execution

- *GetLabConfiguration*

  Gets the lab configuration of a lab server

- *GetLabInfo*

  Gets general information about a lab server

- *GetEffectiveQueueLength*

  checks on the effective queue length of the lab server

- *GetExperimentStatus*

  checks on the status of a previously submitted experiment

- *Cancel*

  Attempts to cancels a previously submitted experiment.

- *Submit*

  Submits an experiment specification to the lab server for execution

- *GetLabStatus*

  Checks on the status of the lab server

- *RetrieveResult*

  Retrieves the results from (or errors generated by) a previously submitted experiment

Client Items

Client items are opaque general purpose fields that belong to a user and are stored in the Service Broker's SQL database. They can be used to store user preferences or details. These are not pass-through methods and the Lab Server is oblivious to their existence. The functionality available to the client is as follows.

- *DeleteClientItem*

  Deletes a client item given its name

- *ListAllClientItems*

  Returns all the users client items

- *LoadClientItem*

  Retrieves a single client item given its name

- *SaveClientItem*

  Adds a client item with the given name

iLab 6.0 added functionality [19]

The release of the 6.0 architecture added several useful functions to the client.

- *RetrieveSpecification*

  Return just the specification of an experiment

- *RetrieveLabConfiguration*

  Retrieves the configuration of the equipment for a particular experiment

- *SaveAnnotation*

  Add a plain text note or description to this experiment

- *RetrieveAnnotation*

  Load the experiments annotation

- *GetExperimentInformation*

  Returns the metadata [everything but the results] for an array of experiments.

  There is only one method that the Lab Server invokes on the Service Broker and that is the *notify()* call. It alerts the Service Broker that an experiment has completed whereafter the Service Broker can alert the user via email.

2.4.3.2 – Java Proxy

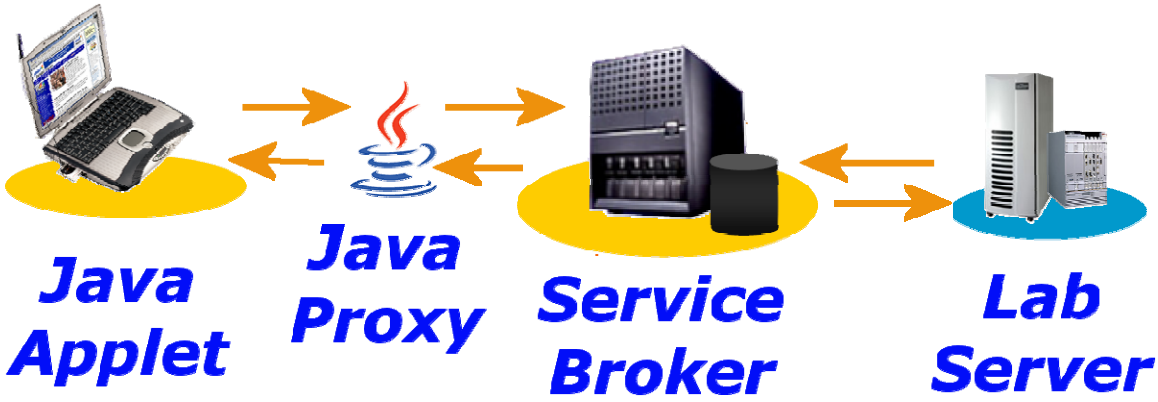


Figure 2.5 - Software layers of iLab with Java Proxy Service Broker Interface

The MIT Microelectronics Weblab project [18] made use of a Java Proxy Service Broker. This required the API to be implemented as a Java class that could invoke the relevant web services on the real Service Broker. This provides for an added level of abstraction for the developer who needs only compile and write code against this class further removing themselves from the details of the web services. Currently only the nine pass-through methods above are implemented, no client item methods or version 6.0 added functionality.

3 – Design Specification

3.1 – Reasons for the choice of the iLab architecture

The iLab architecture was selected due to its high level of scalability and general nature. With the administration, authentication and TCP/IP communication already implemented a significant portion of the work is taken care of. Additionally the scalable architecture conforms to the desire for additional remote laboratories subsequent to this project. Becoming part of a larger community of institutions sharing their lab equipment could also prove beneficial with access to a wider range of experiments and the potential to form research relationships with universities scattered around the globe. The telelabs project at the University of Western Australia has similar goals to the iLab project however it is predominately an in-house research endeavour whereas iLabs is actively pursuing the participation of external institutions. Additionally the MIT iCampus project is more highly funded than telelabs and has a more developed centralised architecture structured for external use.

3.2 – Experiment choices

As this is the introduction of the iLab architecture to UQ and the first of what it is hoped will be many online experiments at this university, this venture is somewhat of a proof of concept exercise and should also inspire interest in online experimentation at UQ. In conjunction with simply demonstrating remote control of equipment, the equipment and experiment itself should be of academic merit and interest and a genuine advantage should be generated by its availability online. Through investigation and liaising with fellow faculty members and staff Dr. Schulz formulated a list of three potential experiments for this project.

3.2.1 - Slotted-line – S-18 Transmission Line Trainer from Signet Electronics. It is a fully automated slotted line that interfaces with a PC using an RS-232 connection. It is capable of operation in several modes each of which represents a different transmission line experiment. Experiments on impedance variation along mismatched lines, impedance matching, insertion loss and spectrum and network analysis can all be conducted by this device.

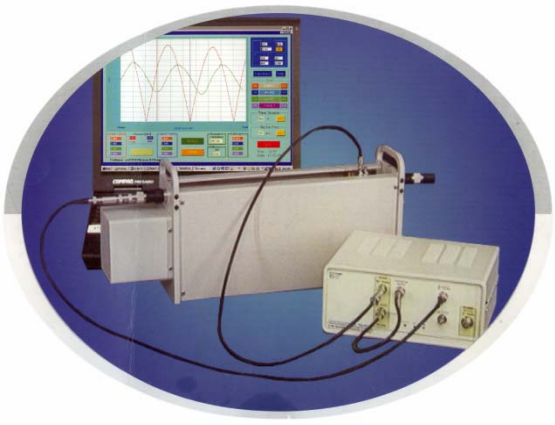


Figure 3.1 – S-18 TLT from Signet Electronics

No experiment of this type is known to be available online anywhere in the world and its addition to the internet would further develop the diversity of remote experimentation. Additionally it is an expensive piece of equipment and hence online usage could help utilise the investment and enables UQ to bring something important to the iLab community. This experiment is also a good candidate as the educational value of this experiment would suffer negligible alteration when operated through a web interface. Being an RF/microwave experiment it is based heavily on theory and of course there is nothing to actually watch and observe. All measurements must be taken through sensors.

Related to this however is this experiment's most important drawback; it is complicated and it does not move. This is a weakness on a PR level and does not diminish at all the academic interest of the equipment. The significance of a slotted-line experiment is generally only understood by electrical engineers, specifically RF and microwave engineers. For the general public and some types of engineers this experiment is difficult to understand and there is nothing to watch in operation. When the experiment is running it looks much as it does when it isn't and it would be difficult for the masses to get excited about. As inaccurate as it may be, people will instantly feel as though something significant has been accomplished if things are moving, lights are flashing and they can understand what is happening. It must stimulate their senses in order to stir their interest in the small window of time you have their attention.

The fact that this experiment doesn't "put on a show" is by no means enough to rule it out as the first online experiment at UQ and for six months this equipment was slated for integration into the iLab architecture. However this experiment choice became increasingly unworkable as time passed. Signet Electronics, whilst founded and run by an academic is a commercial institution and would not release the source code or technical documentation for the slotted-line even for academic purposes. This equipment as an iLab had to be finally abandoned in August 2005 when the device had still not been delivered to UQ.

3.2.2 - AC Machine

The university has purchased several AC motors and their potential use in an iLab project was flagged around the same time as the slotted-line. It is envisioned that this equipment would be used by power engineering students. While uncertainty regarding the use of the slotted-line grew this was kept as the first port of call in the event that the first experiment was dropped. The integration of these machines would have been a substantially larger task as they do not currently interface with a computer. None the less these are large expensive machines of academic interest that would make excellent iLabs. The noise, voltages and moving parts also provide safety benefits in its remote usage.

3.2.3 - Inverted pendulum [20]

Figure 3.2 – Inverted pendulum

This is the experiment that was ultimately chosen to be implemented as an iLab. It is a well known control theory experiment where by control laws are derived and tweaked as to balance a pole much like one balances a broom on their finger. This is a difficult practical run over four weeks in the UQ course METR4202 – Advanced Control and Robotics. The students construct models using Simulink within Matlab on their workstation and compiles and runs their designs on the hardware through connection to a second machine running the xPC kernel. This machine then directly interfaces with the inverted pendulum hardware.

Access to the lab in which the equipment for this experiment resides is limited and students in that course can only use the facilities for a few hours a week during their allotted prac time. Therefore the implementation of this experiment as an iLab would provide a significant increase in its availability to students. This experiment consists of many iterations of design and test. Whilst the mathematical models and computer simulations may successfully balance the pendulum, the actual lab equipment's behaviour is not so ideal. The students could greatly benefit from working with the real equipment whilst they build their early models and perform the many tweaks online, potentially queuing up several variations at once.

Whilst the design and theory required for this experiment is complicated, its actions are simple to understand and when the machine is in operation it is quite interesting to watch and spur on as it attempts to catch the pendulum. This device much like the slotted line is a sophisticated experiment for late year under-graduate engineers; however the goals and results of this experiment can be understood by anyone. In this manner it makes a better showcase for the purposes of creating general interest in the iLab project than the slotted line. Both have real academic applications; the inverted pendulum has charisma. This choice may not expand the pool of online experimentation as greatly as the slotted-line, but it is still an excellent addition.

The decision to use this experiment as the iLab came about as the prospects of the slotted-line were dwindling. A video of the inverted pendulum in action was shown to Dr. Schulz who in turn showed it to Prof. Keniger (Deputy Vice-Chancellor [Academic]). Prof. Keniger then made the decision that it would make a better first iLab than the AC machines. The choice of this experiment was also advantageous as it kept everything in-house. Dr. Hovland designed and wrote the experiment in conjunction with Shane Goodwin and all code and information related to the device was unrestricted. Additionally Dr. Hovland's presence on campus makes him a valuable resource.

3.3 - iLab Specifications

In the development of this iLab it is important that it adheres to the structure, purpose and goals of the iLab project and its parent project iCampus. The development of this iLab must compliment and extend existing practical education by the increased availability of lab equipment and provide for an extendible infrastructure and knowledge base for further expansion of the iLab project through UQ and throughout our region.

3.3.1 – Service Broker

The specifications for the Service Broker come pre-defined and implemented through the iLab framework. The only required manipulation of this software is a graphical redesign for seamless integration with the visual style of the ITEE domain. Whilst no functional changes should be required, the Service Broker's operation and maintenance will need to be investigated as this is the first such machine to be set up at UQ and will provide the base for the development of this iLab and those in the future. It is desirable upon completion of this project to have UQ's first Service Broker in operation on its long-term server machine for

visibility through the internet to the world and to serve as a base for all future iLabs at the conclusion of this project.

3.3.2 – Lab Server

The Lab Server must interface with the equipment as holistically as possible such that as much of the lab experience can be conveyed through to the Lab Client as possible. It should also allow for students to promptly carry out experiments without the need for extensive education in its operation.

3.3.3 – Lab Client

The Lab Client is the face of the iLab project to the student and its characteristics and functionality will mostly directly dictate the success or failure of the iLab in the eyes of these students. In conjunction with the Lab Server, it must present meaningful use of the lab equipment to the students in a manner that compliments and extends their existing education. It must provide as many of the facets of learning present in physical lab access through its interface and its operation must be simple to follow and a pleasure to use. Students should feel that its existence has not only yielded convenience, but has enriched their practical learning by the generation of greater access to the lab equipment.

4 – Prototype Design

4.1 – Service Broker

The most important part of the iLab architecture is that of the Service Broker and naturally this was the entry point for my research. This being the first iLab implemented at UQ and one of only a handful in the world it was somewhat of a trail-blazing exercise. Whilst the setup of the first working Service Broker took some weeks the task can now be undertaken on a bare metal machine within two hours. The instructions supplied by MIT are very comprehensive and generally well written, however they do contain a few small flaws and to the uninitiated these are enough to hinder your implementation. Additionally, as this software is of only very limited release there were some errors and bugs that would be first discovered during the course of this project.

As the Service Broker software comes as a functioning, ready for use release there were no functionality or feature changes required. The research and development of the Service Broker side of the iLab architecture consisted of the creation of experimental Service Brokers for the investigation of their operation and for the development of iLabs. The only significant change made to the Service Broker was an aesthetic one, the graphical redesign of its interface for consistency with the style and layout of the other webpages in the ITEE domain.

4.1.1 – Service Broker creation

For almost the entire duration of this research my own pre-existing infrastructure was used. This provided a more productive platform for development than the issues that would arise from using the universities equipment. This provided a ready-made network of which I was already very familiar and importantly had full ownership of, and administrative privileges to. As an undergraduate, ITEE policy is fairly restrictive on what can and cannot be given access to. The physical convenience of home also made for an excellent development environment.

The Service Broker software would be installed on at least five different machines throughout the project. The first of which was an Athlon 2400+ based Windows 2003 Enterprise machine used as the domain controller and primary server for my small home network. Already running Windows and SQL server it was the logical choice. However the presence of Microsoft Sharepoint Server on this machine quickly proved to be a problem. Although not

documented in any of MIT's release documents there is an issue regarding Sharepoint and the iLab Service Broker software on the same machine in the default configuration. Later it was also observed that the presence of Microsoft Exchange on the same server was also detrimental. The iLab documentation describes the installation of a machine with the Service Broker and possibly a Lab Server as the only IIS webpages. Whilst it makes no mention of the effects of other IIS sites running on the same machine, it appears that this can create problems. These problems usually appear as ASP security or access errors when the Service Broker is accessed.

Two more machines were then employed, both Windows Server 2003 machines on the same network however both without any other web services operating. The intention was to use one as a Lab Server, and the other as a Service Broker. It was here that one of the largest and most mysterious troubles of the project arose. The Service Broker and the Lab Server were not communicating. After several formats and reinstalls, with the Service Broker and Lab Server both on separate machines and or the same machine the problem was still not resolved. The iLab development team at MIT was very helpful and offered some suggestions however they were unable to diagnose the problem. Eventually Service Pack 4 for SQL Server 2000 was installed. The Service Broker documentation outlined that SP3a should be installed; however SP4 was a new release and was as yet untried. As the last step before attempting another format and complete rebuild SP4 was installed and immediately the Lab Server and Service Broker were communicating properly.

Both the team at MIT and myself spent a significant amount of time troubleshooting the problem. The possibility of future users encountering the same problem was reason to not to simply move on once the problem was resolved, but rather locate the source of the fault, or at least attain some detail regarding when it occurs. To investigate the problem several full reinstalls on several different machines under several different flavours of Windows Server 2003 and locale settings were undertaken. The only variation in the installation of the problematic machines from that in the iLab documentation was that these machines had the Windows Server 2003 service pack installed were using Datacenter edition not Enterprise edition. The locale settings of these machines would also be different from those used by the development team at MIT and in the spirit of pursuing all possible differences as potential causes locale settings were also investigated. The list of installation variations attempted is detailed in Table 1.

Windows Server 2003	Locale Settings
Datacenter Edition	English [AUS], AEST GMT+10
Datacenter Edition w/ SP1	English [AUS], AEST GMT+10
Enterprise Edition	English [AUS], AEST GMT+10
Enterprise Edition	English [US], US Eastern Time GMT-5
Enterprise Edition w/ SP1	English [AUS], AEST GMT+10

Table 1: Installation variations investigated in location of Service Broker fault

However in the end all behaved identically; non-functional until the addition of SP4 for SQL Server 2000. After five full system installs the cause of the error was still unknown. It was here that the decision was made that sufficient time had been devoted to the investigation of this issue, and whilst the exact reason for the fault was not established, the issue was reported and a solution found, preventing future users spending considerable time on the issue.

The investigation and resolution of this problem whilst obviously frustrating and confusing resulted in an enhanced knowledge of the underlying operation of the Service Broker that would not have been acquired otherwise. For the development of any single iLab this knowledge is not essential, in fact the layers of abstraction present in the architecture are designed specifically such that you need not be concerned with such details when you are writing Lab Clients and Lab Servers. However this deeper comprehension and familiarity with the inner workings of the Service Broker will be useful as more and more iLabs are created at UQ and the administration and operation of the Service Broker becomes a larger task.

4.1.2 – Graphical redesign

To maintain consistency the graphical design of the Service Broker had to be redesigned to appear fully integrated with the rest of the ITEE domain. The approach taken here was that the page should run off the same Cascading Style Sheets (CSS) as the rest of the ITEE domain. This keeps the visual maintenance of webpage centralised with any changes to the style of the ITEE web carrying over to the Service Broker. The alternative is to simply redesign the look of the page to match the look of the ITEE domain. In that approach the

pages would appear the same, although they are actually completely independent. In the centralised CSS approach the Service Broker looks like the rest of the ITEE domain because it is generated with the same style sheets as they are.

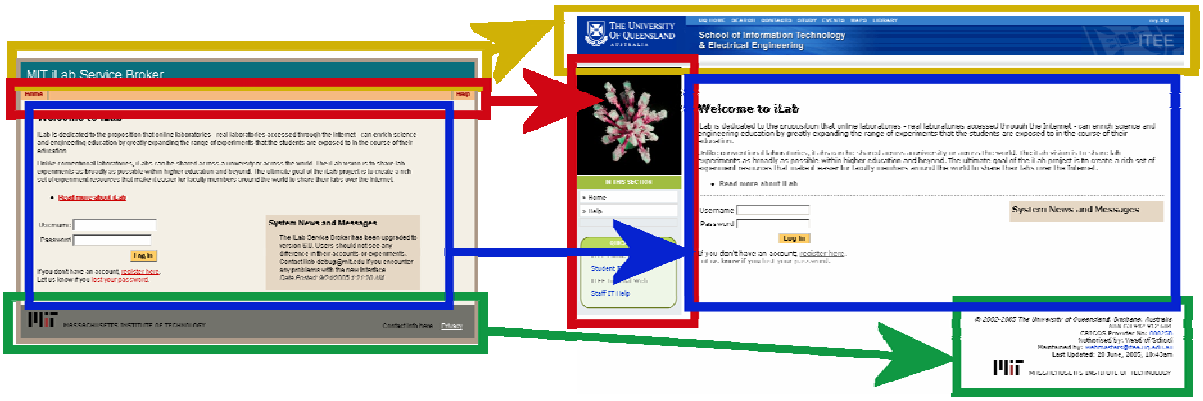


Figure 4.1 – Graphical redesign of Service Broker web interface

In essence the default Service Broker interface out of MIT has a structurally similar design to that of the ITEE domain. That is, banner at the top, navigation bar, content and footer. The most significant difference is the navigation bar that runs horizontally in the default configuration and vertically along the left hand side for the ITEE domain. As I had never worked with cascading style sheets before some experimentation and research in this area had to be undertaken and some familiarity with regards to which tags corresponded to which visual attributes had to be attained. The redesign of the banner, navigation bar and footer to run from the ITEE CSS files rather than the default CSS files was the most complicated component of the graphical change. Once these were created a template HTML page was produced that conformed to the ITEE layout using these newly altered features with a space for the page depended content. For each page in the Service Broker the content was transposed into the template to create the new look page with special care to maintain the functionality of the page through the move. The content section of the page operate off a derivation of the original CSS files that resides on the Service Broker rather than the ITEE maintained style sheets as the ITEE has no specifications for this type of content. The change to the default style sheets consisted of colour and font changes in addition to the removal of any definitions that related to the header, footer or navigation bar that is now controlled by the ITEE sheets.

Once page templates are setup and header, footers and navigation bars are redesigned the page can be updated rather quickly. The original graphical redesign was of the version 5.1 iLab architecture and it took only an hour to graphically redesign the version 6.0 architecture upon its release using our already redesigned components and templates. Unfortunately for other universities or users who will need to redesign their Service Brokers there can be no step-by-step guide or "one size fits all" template on how redesign the style of the pages for your purpose. The diversity of layout and structure in webpages means that each graphical change will have to be considered individually. In a general sense however most pages consist of headers, banners, footers, frames with the page dependent content in the centre. Hence a similar template development process similar to that employed here will likely serve most developers well.

4.2 – Prototypes

4.2.1 – Overview

4.2.1.1 – Lab Servers

The development of the final Lab Server was the culmination of several smaller prototypes, each implementing and investigating different functionalities. Even uncertainty mounted regarding which experiment would be put online, much could still be done investigating how to program and interface the Lab Server and deal with different types of data.

4.2.1.2 – Lab Clients

For each of the four types of Lab Server there was a corresponding Lab Client. This was the most labour intensive component of the three-tiered iLab architecture and arguably the component that would most define the success or failure of the project in the eyes of the users. Beginning with the use and experimentation of the example Time of Day server bundled with the iLab SDK, each Lab Client was built on the previous, culminating in the final Inverted Pendulum Client.

4.2.2 – Time-of-Day (Basic Functionality)

Part of the iLab architecture SDK is an example Lab Server known as the "Time of Day Server". The "experiment" is merely the acquisition of the current time.

4.2.2.1 – Lab Server

The first problem encountered here was the mechanism for acquiring the time was through a query to a time server in the US. For whatever reason, locale settings, firewall or security at the time server end the time could not be acquired in this manner and this unnecessary step of querying external sources for the time was removed and replaced with a simple *DateTime.Now* call. This server was otherwise unproblematic with most of the difficulties at this stage being Service Broker related, although in the early stages of investigation it was not obvious whether the Service Broker, Lab Server or even the client were to blame for the troubles. This example provided an introduction to the development, debugging and administration of a Lab Server. All future servers were derived from this initial example.

4.2.2.2 – Lab Client

This remained essentially unchanged for this functionality, with the only problem for this prototype residing on the Lab Server side as described above. Naturally it was here that familiarity was gained regarding what the Lab Client was capable of and how it was coded.

4.2.3 – Capital City (RS-232)

The investigation of RS-232 communication under the iLab architecture was undertaken with the slotted-line experiment in mind. Even though some uncertainty was growing regarding this as the iLab RS-232 was deemed a likely interfacing requirement in any case. In the end that was correct, however the serial communication was already handled through the xPC API.

4.2.3.1 – Lab Server

This would relay the name of a country submitted from the client through one of its serial ports which was in turn connected to another computer via a null modem. This third computer was representative of the actual lab equipment and would listen for incoming country names and reply on that same serial port with the capital of that country in plain text. In the end nothing was directly used from this Lab Server, however it did serve as an introduction to C# and was the first significant modification of the Time of Day Lab Server.

4.2.3.2 – Lab Client

This client was only a slight modification of the Time of Day Client, with the dropdown box specifying the time format (12 hour or 24 hour) replaced with one containing a list of all the

countries of the world. As stated above this prototype was to implement RS-232 functionality which was not required for the final prototype. However as with the Lab Server, this was the first real modification that was undertaken on the example kit and it was here that the first genuine knowledge of developing for this architecture was acquired.

4.2.4 – Sine Wave Server (Plotting & Graphing results)

As it was clear that regardless of experiment choice, graphing would be involved, a server was implemented that merely fed back two sine wave "signals" of different frequencies to the Lab Client which in turn graphed these results.

4.2.4.1 – Lab Server

This server would eventually act as a pendulum simulator for the development of the final prototype with the two sine waves representing the two angles of the Inverted Pendulum. These sine waves were used by the client before xPC interfacing had been realised and any real data had been acquired. This Lab Server was a trivial piece of software that merely produced two sine waves that ran for some time outlined in the experiment specification field.

4.2.4.2 – Lab Client

Whilst the previous two designs focused primarily on the Lab Server, this was the first client orientated prototype. The user would enter a period of time and the Lab Server would return two sine waves that ran for that time. These results then needed to be plotted.

4.2.4.2.1 – Graphing package choice

It was here that the choice of graphing software was made that would follow through the rest of development. Previous iLabs such as the Dynamic Signal Analyser or Microelectronics Weblab have used graphs however these were small specific libraries built by others at MIT for simple graphs. These were kept in mind however a higher level of complexity was envisioned for the charting in the iLab.

The priorities that would define this decision are as follows approximately in order of importance;

> *Free to distribute* – Most importantly the package must not be bound by any commercial license whereby its distribution through this project would be illegal.

Functionality – The package must be able to plot multiple series, use markers, zoom and adjust axes.
Performance – It must be able to display large amounts of data in a timely fashion
Size – The size must be appropriate for distribution on the internet.
Expandability – Related to functionality. This package should be expandable beyond its current scope of use.
Appearance – Nobody wants to look at an ugly chart. A pretty display makes the client more of a pleasure to use.

The package decision process predominately involved reading through Java programming forums and discussion boards looking for recommendations and reviews with regards to what is available and under what circumstances the respective packager are best employed.

With regards to performance, most of the front runners in Java charting appear to be commercial products such as JViews, Object Planet and Visual Mining often related to plotting stock market and financial information. These packages are expensive and their distribution through the Lab Client may not be legal. Essentially the main free libraries that were most frequently recommended and referenced were JFreechart and Chart2D. From the discussion and reviews online a general comparison of the two was derived.

	JFreechart	Chart2D
Functionality	Excellent	Average
Performance	Average	Poor
Documentation	Excellent	Good
Size	Poor	Good
Appearance	Good	Excellent

Table 2 – Comparison of JFreechart and Chart2D

JFreechart appears to be the most popular of the Java charting libraries and was referenced and recommended more frequently than Chart2D. Importantly it was reported to significantly surpass Chart2D in performance. Whilst JFreechart is not appropriate for refresh rates higher than 1 hertz and is not optimised for real-time charting, Chart2D was given significantly worse reviews in this area and would have rendered it unusable for our purposes. Additionally JFreechart has extensive documentation and a very impressive functionality set.

Its shortfall, which was admittedly overlooked, is its size. The entire package must be contained within the Lab Client jar archive in order for it to run. JFreechart is a mishmash of dependencies where classes completely irrelevant to your implementation are required to be present. An analysis of JFreechart through JDepend quickly showed that minimising the size of the client by removing unnecessary classes was not feasible. Whilst Lab Clients such as the iLab Dynamic signal analyser and the example Time of Day client are around 200kb, it was impossible to create this client below 1.4Mb due to the bloated nature of JFreechart. Jar archives are cached by default so it need only be a one-time download. This is still significantly less to download than labs that require LabView, however this is still inefficient when fewer than 400kb would likely suffice for our functionality set. There are no current plans for the development of JFreechart as a more internet friendly package.

Upon selection of JFreechart its integration into the Lab Client was a matter of reading the documentation and experimenting with its functionality and possibilities.

4.2.5 – Inverted Pendulum [Pole Balancer] Server

Upon selection of this experiment a meeting with Dr. Geir Hovland was arranged and details regarding the experiment operation were discussed. This experiment is a difficult practical run over four weeks in a forth year mechanical engineering course, however many of the experiment details would thankfully not concern me as the iLab developer. The functionality required was explained by Dr Hovland and any questions regarding the operation of the experiment were referred to him.

4.2.5.1 – Lab Server overview

Having had much experience with the iLab architecture by this stage, and with some modules already created in the previous Lab Server prototypes the implementation of the final inverted pendulum server went smoothly. This was due in no small part to the powerful xPC API and its excellent documentation.

4.2.5.2 – Lab Client overview

This client extends the basic charting investigated in the Sine Wave prototype and adds additional features such as a state diagram and pendulum animation. The sine wave client evolved into this prototype with the earlier acting as a source for the animation and charting before and during the development of the xPC interface.

4.2.5.3 – xPC Interface [Matlab]

Having never used Simulink nor the xPC toolkit components of Matlab, the first step was to briefly look into these toolkits and discover what they do and how they can be controlled. The xPC toolkit comes with three methods of operation; it may be run directly from Simulink, manipulated externally using the compiled models and a web interface or by use of a custom program utilising the dynamic linked libraries that come with the package.

4.2.5.3.1 – Consideration of the xPC Web Interface

Initially the web interface was of interest. It possessed many of the features desired such as parameter modification and was already available through a web interface. This interface could in fact be made easily available through the internet as is. However modification of the web interface would be difficult and would involve essentially creating a new web interface that references the original, served from some other location. This would be a very messy and overly complicated implementation with the bottle-neck always at the original web interface. That is, nothing could be implemented that was there originally. All that could be added would be a re-routing of the messages via SOAP through the Service Broker from the new web interface to the old in order to keep it under the iLab architecture. The short-lived investigation into using the web interface component of the xPC toolkit grew out of the recognition that the web interface was already doing at least 50% of what was desired, and that possibly the other features could be added to the already functioning product. However this approach would be very restrictive and cumbersome and was quickly identified as such.

4.2.5.3.2 – xPC API

The xPC API comes in two forms. The standard xPC Target API consists of C functions that can be incorporated into any high-level language application. This would be excellent for building an interface in C or C++. The xPC COM API provides the same functionality but is a suite of interfaces for use with programming environments that work with COM objects. C# is one such language and the documentation and instructions given for use of the COM API with Visual Basic were not difficult to generalise to C#. The use of the xPC COM API proved very powerful and its use was straight forward courtesy of the thorough and easy to follow documentation.

4.2.5.4 – Email notification

The task of email notification is usually the responsibility of the Service Broker under the iLab architecture. For this iLab however the responsibility of notification has been placed at the Lab Server. For the Service Broker, the experimental results are opaque and meaningless. All the Service Broker can do is alert the user that experiment *x* finished at time *y*. The Lab Server on the other hand understands the experimental data and can relay any information you choose through the email notification. For instance, in this implementation the email not only returns the finish time and experiment ID, but also experiment specification details, full explanations of any errors that may have occurred and a brief sentence form summary of how the pendulum behaved during the course of the experiment. Whilst all this information is still available through the Lab Client, having more information in the notification allows the user to make better decisions earlier. For example, in the event of unexpected errors the user may wish to logon again earlier than they planned or shift the task up their daily list of priorities if they were counting on those results. Although not implemented here, if desired certain types of errors could send additional emails to iLab administrators alerting them to potential hardware problems much sooner than the emails of frustrated and confused students. Having a messenger that understands the message allows for a more powerful notification system.

Some redundancy is introduced by this approach however as only the Service Broker knows the users email and in order to relay an email address to the Lab Server it has to be entered into the client and sent with the experiment specification. The general purpose client items were used to remember the users email address. However only the nine pass-through methods for invoking the Lab Server were implemented in the Java Service Broker Proxy class that came with the iLab SDK, and hence the four client item methods had to be written to make use of the client item functionality on the Service Broker. The use of an email address in client items as opposed to the one registered with the Service Broker has additional advantages as the user can received email notification on whatever address they desire, not exclusively at the address they have registered with the Service Broker. Regarding security, placing the Lab Server with the responsibility of email notification requires that port 25 be open. If future iLabs require the Lab Server be visible only to the Service Broker, and Lab Server administered email notification still wishes to be employed, then the Lab Server will have to be configured to pass all emails through the SMTP server at the Service Broker instead of delivering the mail directly.

4.2.5.5 – Webcam web publishing

The webcam is a staple of many online laboratories however as will be shown later, it is made largely redundant by the high-quality animation of results discussed later. However it is necessary for demonstrative purposes and proves to users that this is not simply a simulator, the experiments are actually run on real equipment. As outlined in a survey of students using the Dynamic Signal Analyser iLab at MIT_{15}, students that feel and understand that their experiments are carried out on real equipment rate the effectiveness of the iLab much higher than those that would liken it to a simulator.

A Panasonic colour security camera was used in conjunction with an IP Video 9100A made by AvioSys for the webcam. The 9100A is an embedded web server for viewing composite video through a web interface. The 9100A is an excellent device however it is a commercial product and hence is poorly documented for research and development. The inbuilt web interface could of course simply be made available in its default form straight out of the factory, however customising the look and functionality of the web display makes for a more professional and functional service.

There are three different methods of displaying the video through the web interface;

Active X – This produces the best results and is only for Windows

Java – The slowest method, but for all platforms

MJPEG – Motion Jpeg stream, high bandwidth.

The exclusive use of Active X was immediately discarded to maintain cross-platform support. The original thinking was to make the webcam visible through the Lab Client, keeping everything the user needed to see regarding the experiment on that one applet. However it was the Lab Server that was to become responsible for the webcam. The prospect of viewing the video stream through the Lab Client possessed disadvantages. Firstly, the poor documentation made working with the 9100A for this kind of project difficult. Secondly, the Lab Client typically does not know what experiment is currently running. If desired this could be circumvented by sending back information from the Lab Server regarding the currently running experiment via the *getLabConfig* or *getLabStatus* calls available to the client. Thirdly, the addition of the webcam to the Lab Client could produce unnecessary demands on the CPU and internet bandwidth. To have the webcam administered by the Lab Server meant the webpage could contain as much detail about the current experiment as desired. The drawback of this approach is the Lab Server is now a web server that must be

visible to everyone. If very strict security requirements were in place this could be a problem, however in this case it is not a concern.

The development of the custom web interface involved viewing the HTML source of the default interface and trying to discern how it was acquiring and displaying its video. The Java and MJPEG methods of video display were both comparable in picture quality though the Java video was jerkier than the MJPEG. However the smoothness of the MJPEG came at a high price of 500kb/second; fine for LAN use, not accessible to most internet connections. Jerky video is not a major concern as the animated results replay the experiment at up to 67 frames/second. The user must already have Java to be using the Lab Client and the slow frame rate is the cost of low bandwidth.

To use the Java video a page was created that conformed to the look and layout of the ITEE domain and utilised the ITEE CSS with a space for the video and experiment information. The Java Applet video was then transplanted into the page using modified source from the default web interface. The HTML source for this page was then split into two files, one for each side divided by the position on the page that was desired to print experiment information. Now for the Lab Server to update the page it need only read the two source files, sandwich its message in between and output the result to a HTML file.

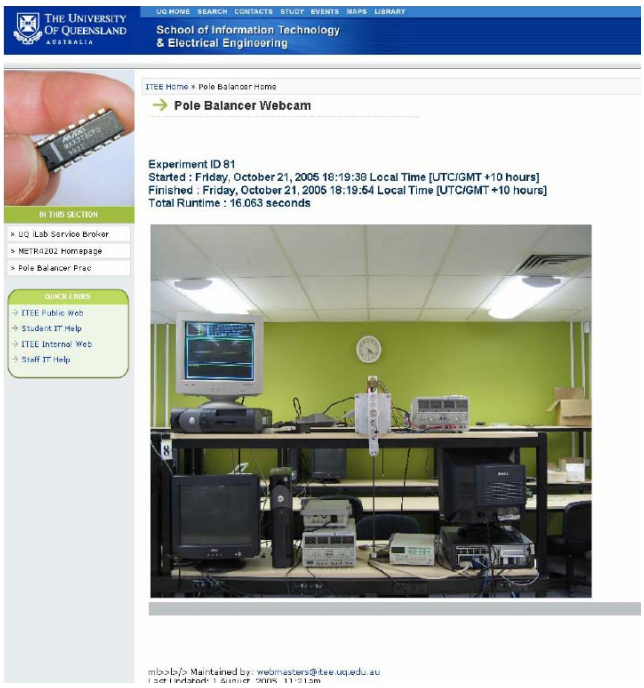

The University Of Queensland
School of Information Technology & Electrical Engineering
ITEE Home » Pole Balancer Home
Pole Balancer Webcam
Experiment ID 81
Started : Friday, October 21, 2005 18:19:38 Local Time [UTC/GMT +10 hours]
Finished : Friday, October 21, 2005 18:19:54 Local Time [UTC/GMT +10 hours]
Total Runtime : 16.063 seconds
IN THIS SECTION
» UQ iLab Service Broker
» METR4202 Homepage
» Pole Balancer Prac
QUICK LINKS
ITEE Public Web
Student IT Help
ITEE Internal Web
Staff IT Help

Figure 4.2 – Webcam webpage

After the Java based webcam was created some further investigation using a similar, “cut, paste, sandwich” approach was extended to an ActiveX based webcam page. That way, if users are using Windows, then they may enjoy the higher quality webcam. This approach however became difficult. Windows, especially XP with SP2 is naturally suspicious of

ActiveX controls and their use was not as straight forward or familiar as for Java. Whilst some inroads were made, Windows would generally block the control when it was not coming directly from the camera and the implementations that were not blocked could not display the video correctly. The use of ActiveX was abandoned. Whilst it does provide higher quality, it was not deemed worthy of extensive investigation, especially as the animation on the Lab Client was designed as much as possible to make it obsolete. The ActiveX link served straight from the camera was made available to users who wished to view it, in addition to the modified Java version with the experimental information and ITEE style.

4.2.5.6 – Simulink file upload

The experiment specification field of an experiment report is simply a string. Hence it has the potential to contain almost any type of data desired as long as it is formatted such that the recipient can understand, and is less than the maximum string length. However there is at least one constraint for the sending of binary files in an experiment specification under the iLab architecture. SOAP is an XML derivative and has special characters and operands. It is not possible to simply place anything in a field as certain characters or combination of characters will confuse the schema. It is a similar situation to defining a string containing the double quotation mark, placing it directly into the string declaration will cause the compiler to believe the string ends at that quotation mark and it will be confused as to the purpose of the subsequent characters.

To avoid this problem the file data has to be converted to some other format. Hex was chosen as it is simple to implement and with a character set size of sixteen it is easy to ensure that no conflicts can exist. Of course it doubles the amount of data that needs to be transmitted. The only alternative that would result in a more efficient file data size would be to convert any special characters or sequence of character that are contained within the file to a different non-conflicting set of characters and decode them back at the Lab Server end. That would prove a messy task and it would be difficult to ensure that all possible conflicts have been accounted for. To save around 30kb for this iLab it simply is not worth the hassle. If for future iLabs very large files (in the order of many megabytes) need to be transmitted as part of the experiment specification, then it is likely that the specification string of the experiment report would not be utilised anyway, rather these large files would circumvent the iLab architecture and post to an offsite address where the Lab Server can download it from. No

iLabs have required such large files to this point, and they are unlikely to require them in the foreseeable future. Hence any pursuit of a more efficient file format is not of any use.

The reading, writing and transmission of binary file data are prerequisites for the implementation of hex data transmission. Hence before the conversion of binary to hex was added, the transmission of file data in binary form was tested to discern just how likely it is that a file will contain illegal character strings. For standard text files containing sentences and words in the order of 10-100kb transmission was successful in all of the five files tested. Although the files were not verified and it is possible the integrity of the data was compromised in the process with such symbols as '&' causing the succeeding character to be interpreted differently. Three files containing programming source code text namely Java, also transmitted and compiled after transmission. C code would likely be a different matter where the '&' symbol is used to reference the memory address of a variable. Of the three binary files each of approximately 40kb, one executable, a zip file and a Simulink .dlm none could be transmitted successfully. The error was always similar and was related to invalid Unicode character references. The character combination '&#' used to define Unicode declarations in XML would occur every 256^2=64kb if all characters are equally likely to be present, providing a high likelihood that even a small file will contain this sequence. The chances of it being terminated with a correct Unicode value and semi-colon are extremely unlikely and of course would corrupt the data in any case.

After the data was converted and transmitted using hex an unforeseen problem was discovered. Being new to C# it was not realised that Java and C# interpret characters differently. Java reads in the characters of a file signed bytes, whilst C# interprets all characters as positive unsigned integers. Subsequent research revealed C# is a poor handler of raw file data. Under .NET it is possible to import Java libraries from the J# development kit within Visual Studio for use within C#. The use of these libraries means that Java is interpreting the data at both ends and hence consistency is maintained. The additional constraint is that any computer that runs this Lab Server must have the *Microsoft J# .NET distributable package 1.1* installed. This is a small prerequisite component of Visual Studio 2003 that can be installed independently.

In a feature unused in this Lab Server, but potentially useful for future iLabs, this server accepts the upload of zip archives for transmitting several files or maintaining folder

structures in a specification. Naturally it also decreases file sizes. This was implemented when it was mistakenly believed that the Simulink .mdl files were needed by the .dlm. The .dlm is already compressed by nature, however the .mdl files are plain text and compress well. This misunderstanding was simply due to insufficient familiarity with Simulink. Any future iLabs that involve file upload especially with directory structures will find this functionality useful.

4.2.4.7 - Experiment validation

The .dlm file is only understood by the Lab Server or more accurately the xPC kernel and hence it is the only machine that can verify its validity. It can check that any specified parameters actually exist, whether the Simulink model conforms to the requirements of this lab and also that the Simulink file itself is valid and non-corrupt. In this experiment though, it is not possible to check the validity without actually interfacing with the xPC kernel and this cannot be done without exclusive access to the xPC machine. If an experiment is running the validation cannot be performed until it stops, at which time all steps short of actually starting the xPC application could be carried out and any errors detected. This is impractical and would significantly slow down the use of the equipment. Every experiment would have its specification sent twice, once for validation and then once for submission if successful. Additionally the validation steps themselves would take at least 15 seconds and these exact same steps will still need to be carried out again when the experiment is actually submitted.

The approach to validation taken here was to focus on robustness. The entry fields on the Lab Client have listeners that remove invalid characters. The parameter specification fields clean up any incorrectly orientated slashes (‘\’ to ‘/’) and will even accept the use of a full stop instead. The submission button will not even be visible on screen if the client detects an invalid specification. At the Lab Server end there is error checking at each step and the generation of any faults will cause the immediate termination of the experiment and a full error summary to be sent by email and printed to the Lab Server’s monitor. The requirements of the Simulink model were kept to a minimum to decrease the likelihood that the user would upload an invalid model. The only constraint on the model is that it must contain a single target scope with a three input multiplexer attached feeding in the state, angle 1 and angle 2 signals in that order. The experiment will still run successfully even if there are multiple scopes just as long as the relevant one has the lowest scope ID. If the signals are entered in the wrong order then this will carry through to a nonsense animation on the Lab Client upon

retrieval. If students upload invalid specifications so frequently as to result in less successful experiments per unit time being completed than are saved by not validating, then this approach may have to be reconsidered. However with an average experiment runtime of around two minutes students would need to be consistently uploading invalid specifications over 15% of the time for this to be a concern. With only small modifications required it is unlikely many users would have difficulties after the first three or four submissions.

4.2.5.8 - Experiment result format

Each "word" in the experiment result is separated by the tab character with the first two stating the sample size and number of samples respectively. Whilst it is true that the sample size is contained within the experiment specification it was not possible to retrieve the specification from the client until the version 6.0 release of the iLab architecture. Even subsequent to this release it is still advantageous to redundantly state the sample size within the experiment result as doing so circumvents the need to make a *getExperimentSpecification* call which could be around 40kb with file data. The sample number also needs to be defined here as it may differ from the value in the specification. The specification sample count represents how long the experiment would run at a maximum. However if the experiment falls into the fail safe state it terminates after three seconds. Hence the actual run time must be recorded in the experiment results.

After these first two elements comes the angle data. The layout of the data consists of the angles truncated to four decimal places, interlaced and separated by the tab character. That is, *angle1[1]* *angle2[1]* *angle1[2]* *angle2[2]*…with each set the sample size apart.

```
<Result>
        <ExperimentID>1</ExperimentID>
        <StatusCode>3</StatusCode>
        <ExperimentResult>15 1750 0.0000 -3.1416 0.0000 -3.1416 ... -0.3447 0.3229 E
         0 0 134 3 201 2 228 1 269 2 322 1 365 2 420 1 466 2 525 1 576 2 641 1 701 2
         777 1 839 5 865 6 X</ExperimentResult>
        <XmlResultExtension />
        <XmlBlobExtension />
        <WarningMessages />
        <ErrorMessage />
        <TimeStamp>17/10/2005 10:55:57 AM</TimeStamp>
  </Result>
```

Figure 4.3 – Example experiment result XML object

The character ‘E’ is then printed to signify the end of the angle data section. This E is unnecessary if the data is valid as the client is told how many samples to expect at the start, however if the data is incorrect the E will be out of place and the client will terminate the loading of the results preventing a possible crash when it overshoots. At the end of the experiment result is the state transition information. Only the state changes are recorded, not every sample which would be grossly inefficient. The sample number is simply stated followed by the new state number. The end of the result record is signified with an X.

4.2.5.9 – GUI Overview

The GUI coding was designed for easy manipulation. Most aspects of the GUI are defined by a few global variables describing borders, spacings between objects, positions, colours, font etc. so that the interface can be graphically altered without delving deeply into the code. The client was also designed to effectively use as much screen space as the user has. In the animation frame the components resize with the client to fill the entire space of the resized window. In the case of pendulum animation it can even be undocked which is particularly useful for users with dual monitors.

4.2.5.10 – Animation of results

The animation of the results was designed to utilise as much or as little of the system resources as desired. If the user has a large screen, or multiple screens the client can make effective use of that added space. However if your screen is small, everything will shrink to fit. Depending on the user’s computing power or preference the animation can be a realistic and photo driven, or a simple stick and ball model. The frame rate can be turned up or down and you can compare as many experiments simultaneously as the computer can handle. Series’ can also be turned on and off at will.

4.2.5.10.1 - Timer

A well known problem in Java is the absence of a high precision timer. Under Linux the Java timer has a resolution of around 1ms. However under Windows XP this it is approximately 15ms. This is a significant problem for time critical programs that require high resolution. The animation on the client however does not require high accuracy. The time is printed on the screen and that time is accurate to the frame of animation the user is currently viewing, even though the animation may be running at slightly faster or slower than real time. Whilst it

is was not critical, some attempt and consideration was made to improve the precision and accuracy of the animation.

With the release of Java 1.5 the long problematic timer resolution problem was partially fixed. The introduction of the *getNanoTime* method provided nanosecond resolution. However upon investigation it appears that the interrupt based timers used in Java 1.5 are still based on the inferior *getCurrentMillisecond* method as no increased accuracy was gained when the client was compiled and used under 1.5. The best timing accuracy was accomplished by using a tight while loop that contained a *getNanoTime* call in its condition statement to check if sufficient time had elapsed, however this was far too CPU intensive even when implemented as a thread with lowest priority. The high resource usage of this approach actually slowed the animation down introducing further inaccuracy. The third approach would have been to use native timers. This would have locked the client for use under a single platform namely Windows, but would provide for at least 1ms resolution. iLabs are supposed to be platform independent and hence that approach would not be appropriate. Additionally including and running dynamic linked libraries from within a compressed jar can be complicated. The final decision was to return to the standard Java timer class approach although to improve accuracy, the sample size options were chosen to correspond with the timer resolution in the worse case, i.e. Windows NT. This is the reason the sample size dropdown box contains only multiples of 15ms. This provides for the best combination of platform compatibility, processing load and accuracy.

4.2.5.10.2 - Chart

The use of the chart on this client has been designed to deal with large datasets. As mentioned earlier, JFreechart is not the highest performance graphing package, and if a graph of a few thousand data point is plotted, even the most powerful machine will struggle. Hence to maximise efficiency the chart only plots what the user can genuinely see. If the user is zoomed in, the client removes all data points from the series' that are not currently visible, and never plots more points than there are pixels on the screen. That is, if you open data that is 10,000 samples, and your screen is 1000 pixels across, only every tenth sample will actually be plotted and a linear 90% saving in processing time is accomplished. The time marker on the chart is only updated every second as unfortunately each time it is repainted, the entire chart is also. The time, zoom and frame rate sliders enable quick and easy

movement to anywhere in the experiment and gives the user as much or as little detail as they desire.

4.2.5.10.3 - State machine

As the state changes very little compared to the angle this data series is transition based as stated above. Reflecting this, the state diagram is only repainted when a change occurs, not every frame as the other animations are. The state diagram was designed in a very general fashion. The arrows signifying the transitions are indexed. A two-dimensional byte array links the arrows to their state transitions. For example to discern what arrow connects state one and two you would reference *arrows[1][2]* or *arrows[2][1]* which would return the index of the arrow joining the states. The state ovals are also defined in arrays such that the addition or removal of states can easily be accomplished and transitions can be easily redefined. The state diagram was designed for streamlined inclusion in a completely different iLab if desired by future developers.

4.2.5.10.4 - Pendulum

A webcam was always to be used such that users can feel and believe that they are controlling real equipment. However the production of a high quality reanimation of the experiment data would ensure that this was its only purpose. The user does not use the webcam to analyse their results and consequently there is no need to archive the video feed from the webcam. Video files are extremely bandwidth intensive and to archive video would rapidly result in a massive amount of data. This would be compounded by the fact that the Service Broker archives every call made by a client. That is, each time experiment results are retrieved, the Service Broker does not simply archive the transaction as "User *x*, retrieved experiment *y* at time *z*" but it records the entire set of experimental data that was sent in the message. This is a rather inefficient aspect of the iLab architecture and one that would cause the size of the SQL database on the Service Broker to quickly spiral out of control if video was involved. Any archiving of video would not only be difficult to program and deal with, but extremely inefficient with regards to bandwidth. The only information the user requires is the angles of the pendulum and the state information at different times throughout the experiment. A video sends a plethora of redundant, static and unnecessary information with each frame such as colours and background objects.

The first step in creating the animated pendulum was to implement a simple “stick and ball” representation. Once the basic engine was developed for generating an animated pendulum from the angle data, it could be further developed to incorporate real photographs of the actual lab and equipment. Using a tripod and digital camera four pictures were taken of the lab and pendulum. In each photo the pendulum was static but in a different position. From these four pictures a composite was created and touched up in Photoshop that did not contain the pendulum arms, only the back plate, power supply and the surrounding lab. This was to be used as a static background with the animated arms of the pendulum superimposed on top. Close up pictures of the arms of the pendulum were then taken and the surrounding features in these photos cut out, including those visible through the holes in the driving arm. These were saved as GIF files as they support transparency and are also supported by Java. PNG files are also supported however their lossless nature makes them less desirable for internet usage.

Figure 4.4 – Production of composite background image

4.2.5.11 - Comparison of experiments

The final feature added related to animation was the ability to load and compare multiple experiments concurrently. This is useful as students can load their results and compare them with Dr Hovland's solution or contrast their own experiments with each other. This can give the user a grasp of the statistical variation of the experiment. Even when comparing results generated from an identical specification there is noticeable variation.

In order to maintain speed and minimise processing time, padded two-dimensional arrays were used. When a dataset is loaded it is padded out to remain consistent with the highest resolution dataset currently open and further padded to the runtime of the longest running dataset. If it is itself the highest resolution or longest running dataset, all other sets will be padded. For example if series one has 1000 samples at 15ms each (15 second runtime) and series two is loaded which contains 1000 samples at 30ms each (30 second runtime), then both series will be padded out to create two series of 2000 samples at 15ms each. Series one will have its final recorded value in every entry from the end of the measured data until the 2000th sample. Series two will have every sample appear twice, simulating a 15ms sample time.

Native Series 1 [1000 samples of 15ms]

1	*2*	*3*	*4*	*5*

Padded Series 1 [2000 samples of 15ms]

1	2	3	4	5	5	5	5	5	5

Padded Series 2 [2000 samples of 15ms]

1	1	2	2	3	3	4	4	5	5

Native Series 2 [1000 samples of 30ms]

1	*2*	*3*	*4*	*5*

Figure 4.5 – Padding of datasets to maintain speed and ease of use

Whilst this is memory inefficient it results in far less demand on the CPU. There is further memory inefficiency as the experimental data is stored as both an ArrayList and as a two dimensional array. The ArrayList is used to store all data series in their native format and is used only during the loading of experiments for rapid addition of new datasets. However when an experiment is loaded and all relevant calculations are undertaken the ArrayList is parsed to an array to maintain speed during the animation process. After all an entire experiment of 2000 samples only measures around 32kb, hardly a significant amount for even the most outdated of machines.

However to ensure whatever CPU gains are made by the use of arrays is not negated by the increased number of data points several steps were taken. The native attributes of all datasets are stored so that information is not lost and can be used to make meaningful decisions.

Naturally we do not want to be plotting or animating every point on a padded data set. As it is padded and not interpolated there will be a sudden step change at the point where the padding ends and the next genuine sample starts. Additionally these padded samples are redundant and are only present to keep a consistent sample size and for ease of indexing and reference in an array format. Hence there is an array variable *nativeMultiplier* which defines the number of times larger each series' native sample size is compared with the highest resolution dataset. Then using this and modular division only one point is plotted and animated for each sample.

4.2.5.12 – Feedback Form

A panel was added to the client containing a list of questions regarding the student's iLab experience (*see appendix A for survey form).* This was implemented using the client item functionality and was designed to provide a mechanism for measuring the perceived effectiveness of the iLab in the eyes of the students. Unfortunately as the inverted pendulum a was only available for general student usage for five days and with the deadline for the prac after the deadline for this document a comprehensive study was not possible for this year's class. Few students actually had an opportunity to use the iLab and none completed the survey despite emails reminding them to do so. The first email sent to students to alert them of the new labs availability came with the user manual for the iLab attached (appendix B). Whilst operation of the iLab is simple, the manual is comprehensive and its size likely scared students away from its usage. When no students had utilised the iLab after the first weekend a second email was sent that broke down operation of the online experiment into four steps. Consequently later that day several students logged on and used the equipment.

4.2.5.13 - Integration into the ITS/ITEE infrastructure

Upon completion of this project it was desirable to have the Service Broker and Lab Server installed on long-term machines in their permanent configuration for availability through the internet. This entailed installing the Service Broker on a proper server-grade machine rack-mounted in one of the server rooms. However the complexity of creating a web server at UQ was under-estimated and compounded in difficulty by my status as an undergraduate making me ineligible for any administrative rights on the network.

The network infrastructure of UQ consists of several domains for each of the different faculties. The ITEE is one of these faculties and the ITEE domain itself is subdivided into the

student undergraduate network and the staff/postgraduate network. Originally it was planned to place the Service Broker in the DMZ of the ITEE domain as a web server. An attempt was made to install the Service Broker software on a web server currently in operation in the ITEE domain. However this server was running Sharepoint and the Service Broker's incompatibility with this software was confirmed. The second attempt consisted of installing the software on a server running Microsoft Exchange, again this was problematic. For the third attempt a machine was found and set aside exclusively to run the Service Broker. This installation was without issues however it was not realised and was somewhat surprising that the DMZ of the ITEE can only be seen by the outside world and the staff network, not the students. This was clearly not appropriate as students would not be able to access the Service Broker whilst on campus. The staff in the ITEE infrastructure did not fully understand the full nature of what was attempting to be done and hence had not flagged this issue.

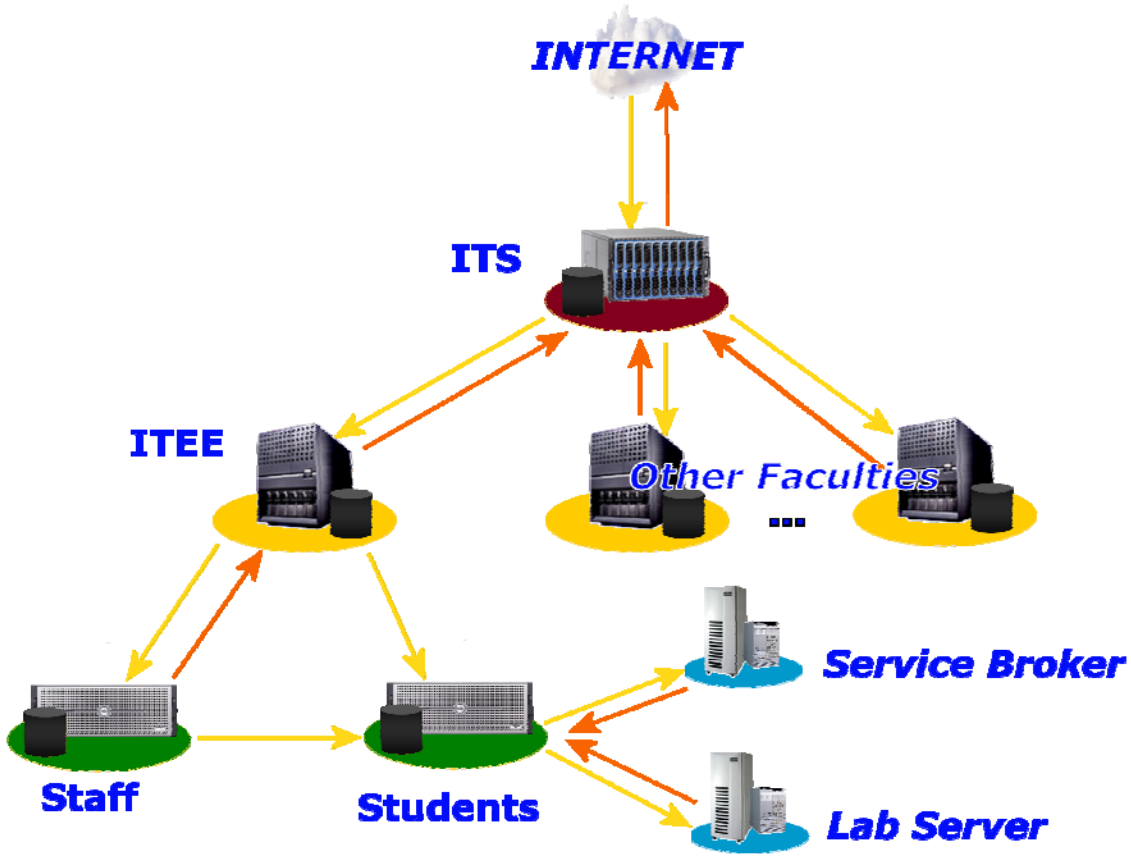


Figure 4.6 – UQ network infrastructure

The student network of the ITEE is isolated from all other networks and can only see the internet through a proxy server. Nothing from the outside can see anything on the student network and the student network can not see the other internal ITEE networks. The student

network is heavily restricted and the processes required to make something visible to that network are high-level administrator tasks that I cannot undertake. Through liaisons with ITEE and ITS administrators it appears that the Service Broker may need to be served from the ITS domain. However the addition of a web server to that network is a long process that requires security plans, budgets and form signing all of which is beyond my scope.

5 – Product Evaluation

The development of this iLab was greatly hindered by problems relating to the slotted-line that were unexpected and beyond control. The inverted pendulum experiment choice was finally made on the 16th August 2005, leaving only two months to create a working iLab prototype. Initially the project was going to operate a slimmed down version of the current inverted pendulum prac in METR4202 involving only the variation of parameters on an existing model, rather than the upload of the student's own. The iLab was also intended for use only by the future METR4202 class of 2006 and subsequent years. However with time permitting the ability to attach models to the experiment specification was also added and the students were briefly given the opportunity to use the iLab for their current prac not specifically written for online usage. This implies that the goal of bringing as much of the physical lab experience to the user as possible is at least partially satisfied. Students have the ability to all the steps possible in a physical laboratory with only minor additional education regarding its operation required.

The nature of this project is such that it is difficult to gauge its effectiveness with hard numerical metrics corresponding to performance. Rather assessment of an iLab is best undertaken through the distribution of surveys and talking to students and staff members. The feedback form was an attempt to gauge the student's feelings towards online experimentation however with limited student exposure a meaningful survey was not possible. Additionally as the iLab was only available for use on campus through the ITEE student network one of the most convenient features, home usage, was not available and hence students were not experiencing the full potential of online experimentation.

As no students filled out the feedback form before the submission of this document their usage behaviour was analysed through the various entries and logs in the Service Broker's SQL database. From the data that could be acquired it was found that home access was likely the most popular usage mode. A total of 6 unique students from a class of 33 logged on for a combined total of 13 hours during the Monday and Tuesday before the inverted pendulum was to be assessed. The average session time was 53 minutes and an experiment was conducted on average every 17 minutes. A total of 46 experiments were conducted with 78% of these performed outside of business hours. The entirety of late night experimentation

between 10pm and 2am was conducted by students with Asian last names, possibly overseas students living on campus. Clearly the aspect of online experimentation of most interest to students is the physical convenience of operating lab equipment from home.

Whilst some small security related bugs had to be addressed subsequent to some emails from students it is worth noting that only one experiment out of the forty-six generated errors. This was produced when the student accidentally uploaded the Simulink .MDL file rather than the compiled .DLM file. This illustrates that the students found the operation of the iLab simple and easy to follow.

By analysis of the records it was also clear that students were utilising the animation to view the results rather than the webcam. With so few concurrent users the experiment would commence immediately upon the users pressing “run” and would have been visible through the webcam. Yet for each of the 46 experiments the results were requested by the client at least once suggesting the students preferred the animation panel of the Lab Client to viewing the webcam. A strange pattern of student usage was also noticed; only one student utilised the email notification system. The others must have either continually pressed “Get Experiment Status” or “Load Experiment” on the lab management panel of the Lab Client until the experiment became available, or viewed the experiment through the webcam to discern when it had completed.

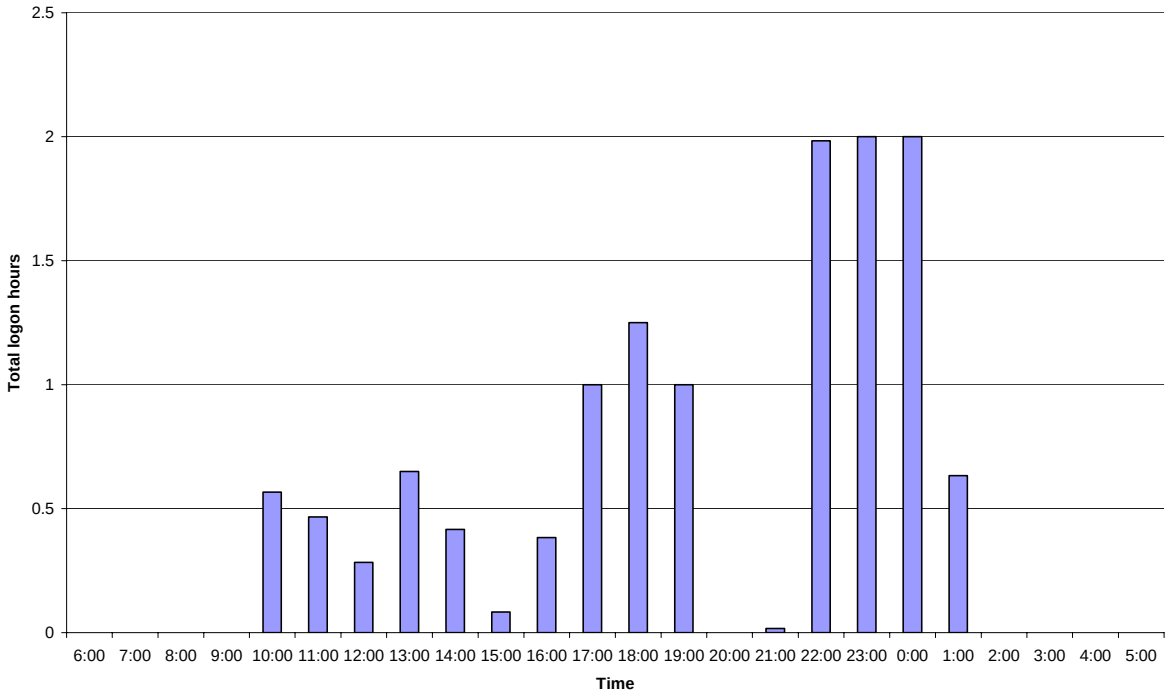


Figure 5.1 – Time spend logged onto Service Broker by hour

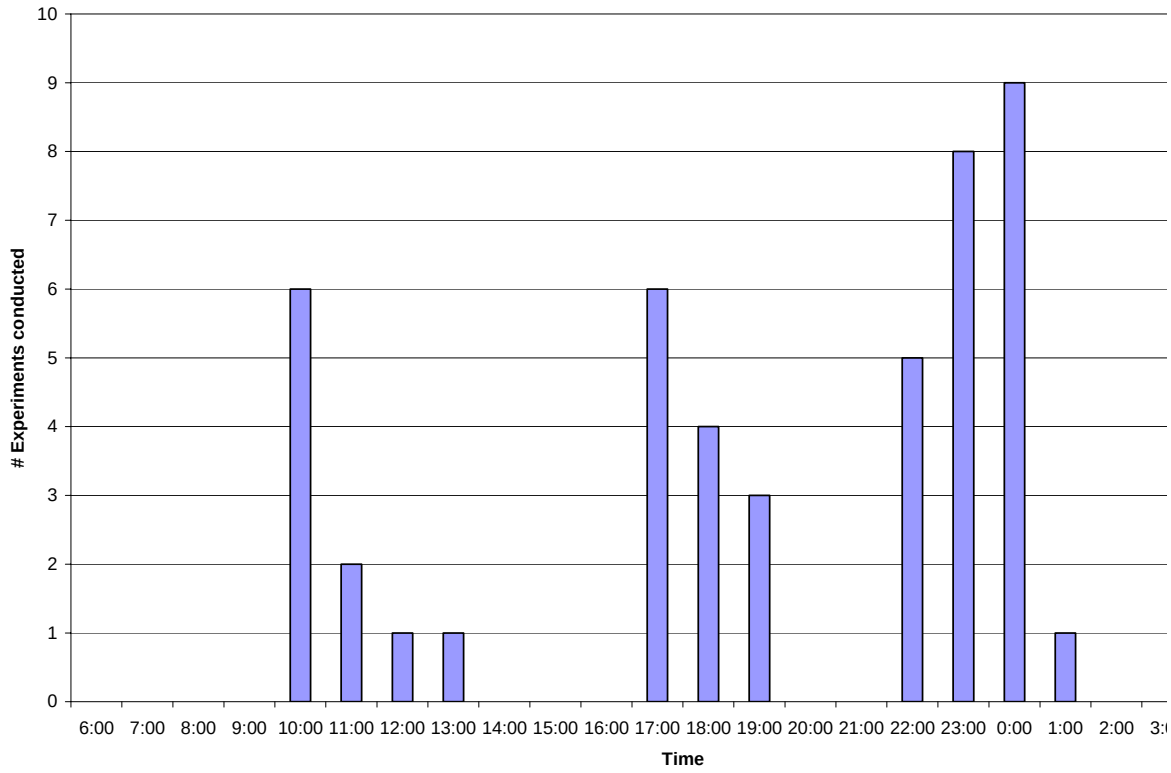


Figure 5.2 – Experiments conducted per hour of the day

In a somewhat informal assessment of the student's interest through conversations and explanations of the project whilst developing in the lab, there was a fair degree of interest in the project. The most interested of the student's were generally those having the most trouble completing the prac on-time and naturally recognised its ability to assist them in completing the prac before the deadline. Unofficial discussions with Dr. Hovland conveyed that he was pleased with the finished iLab and was particularly impressed with the re-animation of real data using actual photographs of the equipment. Dr. Hovland was also the largest user, logging on for a total of 3 hours and 5 minutes to conduct 20 experiments, fine tuning and demonstrating the working solution to the inverted pendulum experiment.

For next year's METR4202 class a demonstration of how to use the iLab should be given during one of the lectures in the first week the students undertake this experiment. This would serve both to inform the students on how to operate the equipment and additionally raise awareness of its existence. The survey form should be mandatory to complete by assessment day.

This project brought together some fairly diverse skills I have not used extensively in my engineering studies. Having never worked with programming web services, C#, XML or CSS this was all new territory. Upon selection of this thesis topic it was envisioned that there would be a significant hardware interfacing focus for this project consistent with other university courses for a Computer Systems Engineer. However the most extensive task in this project was the creation of a high-quality Java Lab Client. Fortunately my early engineering studies were undertaken with a Software specialisation in mind. Another unexpected aspect was the project's utilisation of incidental education obtained externally of university. My job as a part-time network administrator of Windows based servers provided valuable knowledge for this project. The learning curve would have been greatly steepened and problems significantly compounded if in conjunction with familiarising with the iLabs architecture I had to concurrently educate myself regarding Windows and SQL Server networks. Even my background in video production proved useful with skills in graphics and animation proving useful for the Lab Client and the visual redesign of the Service Broker. The main regret of the project was the lost time induced by the troubles with the slotted-line. If the inverted pendulum had been chosen months earlier it would have been possible to have it online ready for students to use for their METR4202 course in the second semester. On the IT infrastructure side, as an undergraduate with no administrative privileges I did all that could

have been done in this area. However if the ITEE and ITS networks had been researched earlier and my requirements submitted to them in first semester then the relevant staff and administrators could potentially have created an appropriately configured machine for use. The convoluted architecture of the university network was unexpected and was overlooked as a considerable problem.

6 – Future Developments

6.1 – Integration into the ITEE or ITS domain

The IT administration requirements for the creation of a UQ iLab Service Broker will require meetings and agreements between ITEE and ITS administrators and relevant staff such as Dr. Schulz. This is a task that will have to occur over the summer break 2005/2006.

In addition to the infrastructure requirements the Service Broker will need SQL scripts or the like to synchronise student details from the ITEE or ITS domain. That is, as the Service Broker handles its own authentication and administration its database will need to be kept consist with the rest of the domain. Students login names and passwords from the ITEE or ITS will have to be continually imported and updated through the use of scripting and scheduling. Ideally the scripts would automatically add students to course groups they are part of that have access to iLabs. Again this is a task for a high-level administrator as obviously they will need to be dealing directly with the usernames and passwords of the entire domain.

6.2 – Addition of iLab 6.0 functionality

The 6.0 release of the iLab architecture introduced five new client methods. *RetrieveSpecification, RetrieveLabConfiguration, SaveAnnotation, RetrieveAnnotation,* and *GetExperimentInformation*. These methods add some desirable functionality to the Lab Client. Under previous architectures the experiment ID was the only field that could be used to identify an experiment, and annotations could only be modified and viewed through the *My Experiments* section of the Service Broker. These features would enable more legible information regarding experiments to be made visible on the client. Rather than using the experiment ID the user could select from a list of their experiments with annotations and experiment specification details to distinguish and describe them. These features are not implemented in the current 5.0 release of the Java Service Broker interface class and would have to be implemented if the developer wished to use a Java client.

6.3 – Magnetic dampening of pendulum

The inverted pendulum must wait between 60 and 90 seconds after each experiment while the driving arm is placed back to the zero-position and the free moving pendulum ceases to

swing. This procedure takes longer than most experiments and very much hinders how quickly a queue can move and the speed at which users can retrieve their results. Whilst it is not possible to put any driving forces on the free moving arm to force it to the zero position, it does contain a steel weight on the end. If an electromagnet was placed below the zero position it could be turned on between experiments quickly dampening the pendulum and significantly increasing the potential throughput of the iLab.

6.4 – Lab Server Broker

A feature not currently implemented under the iLab architecture is the concept of a Lab Server Broker. This machine would be placed between the Service Broker and a Lab Server farm. This could be used to distribute jobs between several Lab Servers running the same experiment creating shorter queue times and maximising the use of equipment. For example, with the inverted pendulum experiment used for this iLab there were five pendulums available. To put all of them online, say over the weekend or after hours, they would have had to be registered on the Service Broker as completely separate experiments. The user would have to remember or be notified by email as to which pendulum they used in order to retrieve their results. A Lab Server Broker would accept incoming jobs from the Service Broker and pass them on to available machines. This does not require a restructuring of the iLab architecture and could be implemented under the current schema. The Service Broker would be oblivious to the fact the experiment jobs are being redistributed and would only retrieve the results from the central Lab Server Broker. A simple way to implement this may be to install both Service Broker and Lab Server software onto the Lab Server Broker. The main Service Broker passes the message to the Lab Server software running on the Lab Server Broker. The Lab Server Broker then passes the messages to a modified Service Broker it has running in addition to its Lab Server software. This “Service Broker” then communicates with each individual identical piece of hardware much like a normal Service Broker treats its many experiment types.

6.5 – SMS Notification

The addition of SMS notification, combined with detailed experiment summaries would prove most beneficial. The user would be able to receive notification even when they are not at their computer, and based on the experiment summary, decide whether they need to view the results immediately, run the experiment again or perhaps that it can wait until later. GSM modems are inexpensive and their operation in C# is not dissimilar to the sending of email.

6.6 – Interactive Experiments

The next stage of development for the iLab architecture pertains to interactive experiments. This will present a different a different set of challenges for the developer but also a potentially more though iLab experience.

6.5.1 – Real-time Graphing

As state previously, the graphing package use by this and those used by other iLabs may not be appropriate where rapid redrawing of the graph is required. If the data rate and screen refresh rate required is low then standard charting utilities could be used, however if future experiment attempt to feedback data from such equipment as oscilloscopes or other devices where large quantities of data can be displayed and changed very rapidly this could introduce problems.

6.5.2 – Real-time Control

Experiments that require quick reaction from the user could pose problems. If the information required to convey to the user the state of the equipment is small then lag times in the vicinity of half of second are feasible using a broadband connection. However if the experiment state is complicated, the internet connection slow and especially if a video feed is involved delay times could rise to a number of seconds and potentially make the experiment unusable. As a video feed would introduce the largest increase in delay as well as other problems it is recommended that any future iLabs that require the user to react in a timely fashion use sensor based data and events not video.

6.7 – HTML Client

The MIT iLab development team plan at an unspecified date in the future to develop a C# Service Broker proxy class similar to the one already developed for Java. When this is released, or implemented by future developers outside of MIT its use would bring added convenience to iLab usage. With HTML clients rather than Java the user can access their iLabs from any computer that has a web browser no Java Virtual Machine required. However to go from a very powerful and easy to use language such as Java, to ASP and HTML could limit functionality or at least increase the difficulty of its implementation.

6.8 – Expansion of experiment diversity

This is the most important course of future development. In order for the iLab project to be successful, productive and alluring to universities it needs to have a wide array of experiments and experiment types within its community. In the short-term the focus should be on putting experiments online that are expensive, of limited accessibility and unlike any others currently available for remote usage. Whilst the slotted-line experiment did not eventuate as the experiment for this project, it is none the less an excellent candidate for iLab usage. The iLab project also needs to be introduced to as wide a range of educational institutions as possible to further expand the effectiveness of the project. The University of Queensland as a hub institution within the iLab project is responsible for the spread of iLabs throughout our region.

7 – Conclusions

An online laboratory was developed that allows students to access the inverted pendulum experiment for use in METR4202. The iLab was designed such that no modification of the prac outline was required. Students have access to the same visual information available in the physical lab both through a webcam and the animation of the results. Additionally they have further capabilities to analyse, inspect and compare their data through the Lab Client which did not exist prior to the creation of the iLab. The experiment is currently available online with the student ITEE network, however further IT administrative tasks will need to be performed to publish the iLab Service Broker to the internet. This online experiment compliments the student's physical laboratory experience and will provide greater convenience and opportunities to learn for the METR4202 class of 2006.

Furthermore an on campus knowledge source has been created by the development of this iLab and the investigation of the Service Broker's operation. This has provided a base of experience with the architecture that is already assisting students commencing iLab related undergraduate and postgraduate research.

8 – References

[1] National Instruments, 2005, *Distance-Learning Remote Laboratories using LabView* [Online] Available at: http://zone.ni.com/devzone/conceptd.nsf/webmain/7BD0B01FCF3CF61A86256B510059F0FB/$File/WP2238.pdf

[2] National Instruments, 2005, *What is LabView?* [Online] http://www.ni.com/labview/whatis/

3. Stanford University, *Cyberlab Experiment Description* [Online] http://www.stanford.edu/~jcsabat/ee366/ex_desc.html

[4] Trevelyan, James, 2004, 'Lessons learned from 10 years experience with remote laboratories', *International Conference on Engineering Education and Research*, Available at: http://telerobot.mech.uwa.edu.au/Information/Trevelyan-INEER-2004.pdf

[5] "iLabs Architecture Overview", iLabs Architecture, MIT [Online] http://icampus.mit.edu/iLabs/Architecture/default.aspx

[6] "Remote Labs on the Internet around the world", Telelabs Project Links, University of Western Australia [Online] http://telerobot.mech.uwa.edu.au/links.html

[7] V.J Harward et al, "The Challenge of Building Internet Accessible Labs (draft)", iLabs Architecture Downloads [Online] http://icampus.mit.edu/iLabs/Architecture/downloads/protectedfiles/ILabChallenge.doc

[8] "iLabs Architecture Downloads", iLabs Architecture, MIT [Online] http://icampus.mit.edu/iLabs/Architecture/Downloads/default.aspx

[9] V.J Harward et al., "Service Broker", iLabs February Release (5.0), SDK compressed archive, MIT, 2004 [Online] http://icampus.mit.edu/iLabs/architecture/downloads/downloadFile.aspx?id=28

[10] V.J Harward et al., "Time of Day Sample Lab Server", iLabs February Release (5.0), SDK compressed archive, MIT, 2004 [Online] http://icampus.mit.edu/iLabs/architecture/downloads/downloadFile.aspx?id=26

[11] V. J. Harward, C. Felknor , *Service Broker to Lab Server API*, MIT, 2005 [Online] http://icampus.mit.edu/iLabs/architecture/downloads/downloadFile.aspx?id=20

[12] D. Zych, C. Felknor, *Client to Service Broker API*, MIT, 2005 [Online] http://icampus.mit.edu/iLabs/architecture/downloads/downloadFile.aspx?id=19 (current April 2005)

[13] "iLab: Remote online laboratories", iCampus Projects, MIT, 2004 http://icampus.mit.edu/projects/iLab.shtml

[14] J.A del Alamo, "iLabs: Performing Laboratory Experiments Across Continents," LINC Workshop, MIT, March 24-25, 2004 [Online] http://www-mtl.mit.edu/~alamo/publications/ 6-28-04 pubs/del Alamo LINC 2004 viewgraphs l.pdf (current April 2005)

[15] G. V. Nuñez, *Design and Implementation of the Feedback Systems Web Laboratory*, Master of Engineering thesis [Electrical and Computer Science], MIT, USA, 2005 [Online] http://www.mit.edu/~6.302/weblab/docs/viedma-meng.pdf

[16] J. Harward, *MIT iLabs*, Carnegie Initiation Meeting, Makerere University, USA, June 26th, 2005

[17] "Extensible Markup Language (XML)", World Wide Web Consortium, 2005, [Online] http://www.w3.org/XML/

[18] "Microelectronics Weblab Documentation", J.A. del Alamo, et. al., MIT, USA http://weblab2.mit.edu/docs/weblab/v6.1/manual/

[19] C. Felknor, *"iLabs Release 6.0 Release Notes"*, MIT, 2005 [Online] *http://icampus.mit.edu/iLabs/Architecture/downloads/downloadFile.aspx?id=46*

[20] G. Hovland, Shane Goodwin, *"Swing-up and balancing control design for inverted pendulum"*, METR4202 Course Material, 2005, University of Queensland [Online] http://www.itee.uq.edu.au/~metr4202/_doc/METR4202-7202-Pendulum.pdf

APPENDICIES

APPENDIX A

Student feedback form created for METR4202 iLab. Form was implemented into a panel on the Lab Client

Effectiveness in teaching course material

- ☐ Very poor
- ☐ Poor
- ☐ Average
- ☐ Good
- ☐ Excellent

Comments

User Interface

- ☐ Very poor
- ☐ Poor
- ☐ Average
- ☐ Good
- ☐ Excellent

Comments:

If you were to do this experiment again, would you use iLab?

- ☐ It'd be a cold day in hell
- ☐ Probably not
- ☐ Maybe
- ☐ Likely
- ☐ Definitely

Comments

iLAB vs. PHYSICAL LAB PRESENCE (*How does this iLab compare with actually being in the lab.)*

Ability to acquire good experimental results

- ☐ Much worse
- ☐ Not well
- ☐ Moderately
- ☐ Comparable to the being there
- ☐ Better than the real thing

Comments:

Ability to learn from the results you acquire

- ☐ Much worse
- ☐ Not well
- ☐ Moderately
- ☐ Comparable to the being there
- ☐ Better than the being there

Comments

iLab Student Feedback Form

METR4202 Inverted Pendulum Experiment

Ability to analyse your results

- ☐ Much worse
- ☐ Not well
- ☐ Moderately
- ☐ Comparable to being there
- ☐ Better than being there

Comments

Convenience

- ☐ This iLab was nothing but trouble
- ☐ More trouble than it's worth
- ☐ No more convenient than the lab
- ☐ Made lab more accessible
- ☐ Made lab substantially more accessible

Comments

How did this iLab effect the time it took to complete this prac?

- ☐ Much Slower
- ☐ Slower
- ☐ About the same
- ☐ Faster
- ☐ Much Faster

Comments

What was the biggest advantage of this iLab for you?

What was the biggest drawback of this iLab for you?

Do you have any suggestions for future iLabs in this subject or in others?

Can you think of another experiment, prac or piece of equipment from a different course that you believe would be a good candidate to go online as an iLab?

APPENDIX B

METR4202 iLab user manual distributed to students

Before you start…

To use this iLab you will need ;

Java 1.5
(untested but probably works on 1.2+)
https://sdlcweb2d.sun.com/ECom/EComActionServlet;jsessionid=E35503775DC0BB15B86C82B37467088B

Matlab 7.01 (if you wish to use your own simulink models)
(the version on the uni computers, later versions will not work & earlier versions are untested)

Username & password

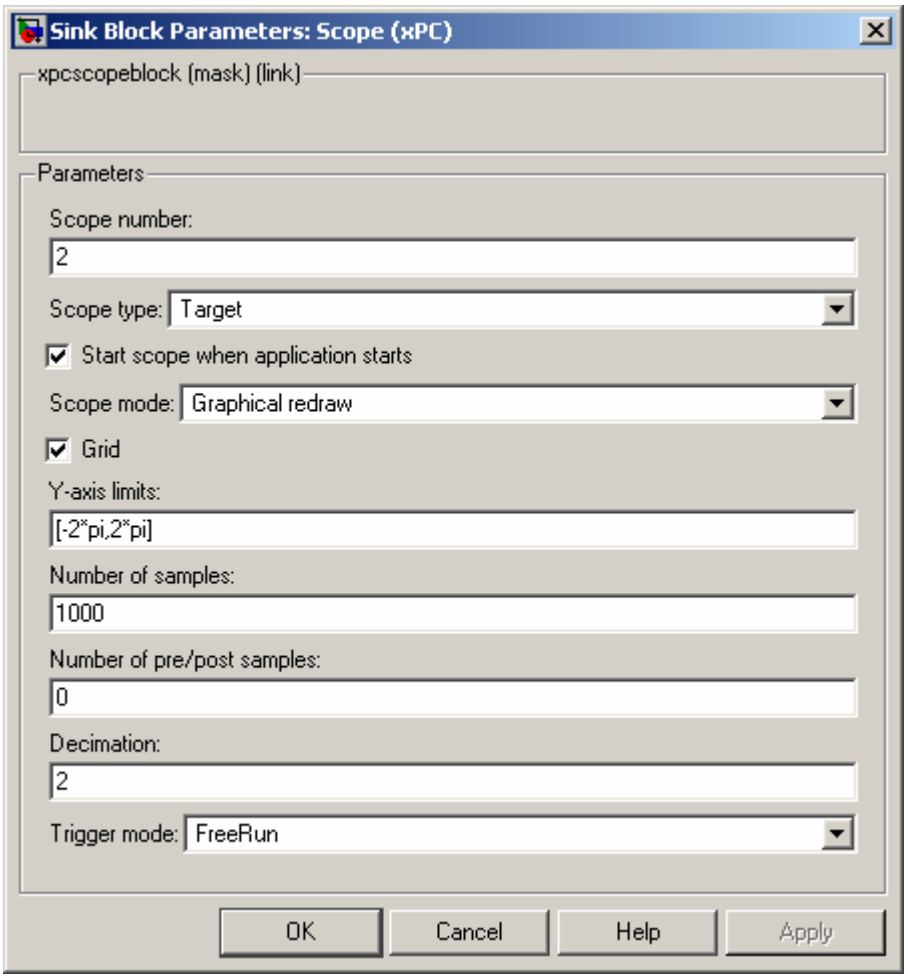
A computer connected to the ITEE network
(the server is only visible within the ITEE network, hence you cannot access it off campus)

Will of steel, possibly eye of the tiger
(that pendulum ain't gonna balance itself)

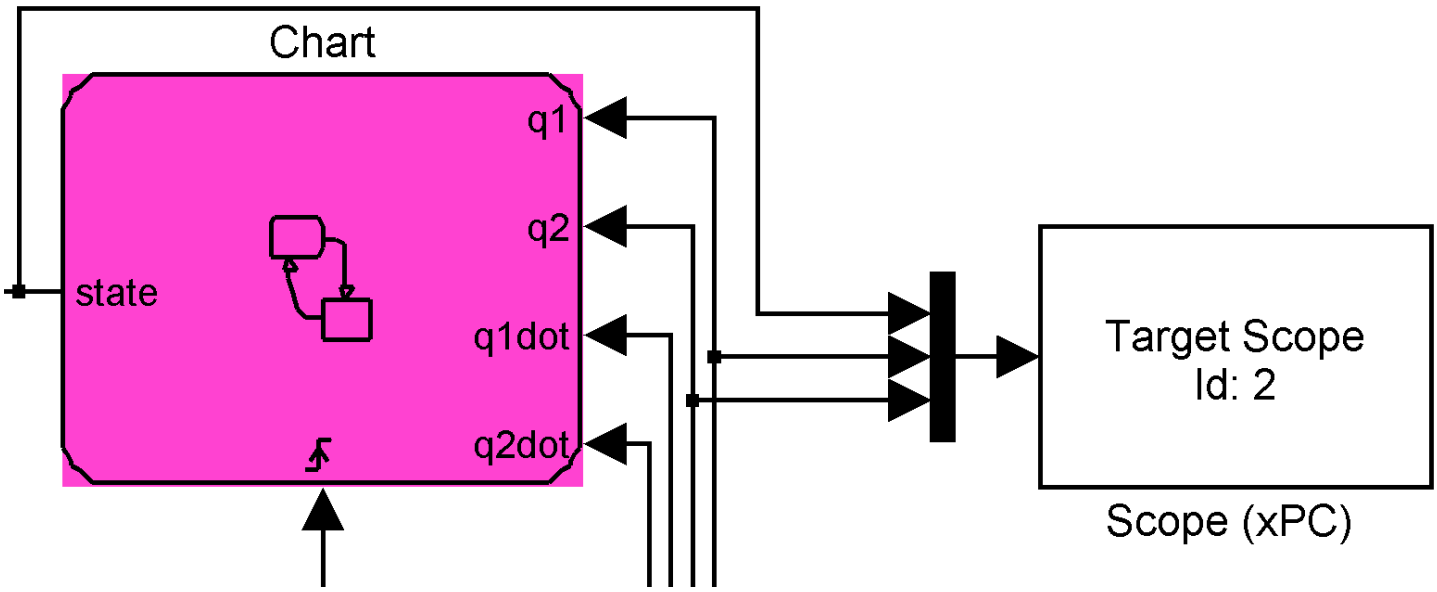
SIMULINK

In order to use this iLab you must have a simulink model that conforms to a few details.

Firstly, you must only have ONE SCOPE in your model. It must be a TARGET scope with three inputs through a MUX as the diagrams below demonstrate. The order you put the signals into the mux is important, it must be; port 1 = state, port 2 = q1, port 3 = q2



Target Scope details, the only parameter to note it Scope Type = Target



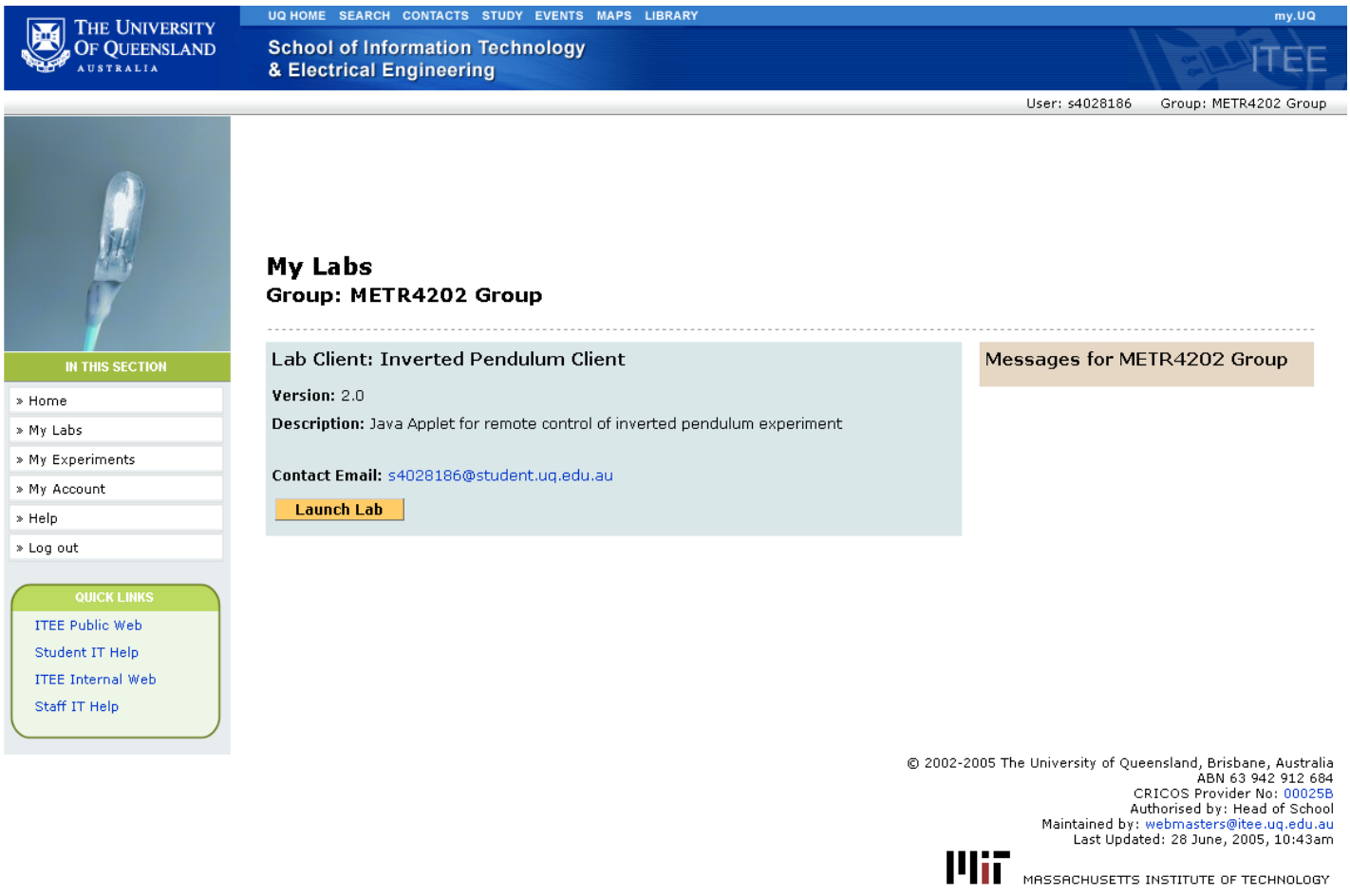
Our target scope with a 3-to-1 mux

Compile the model in the usual fashion with the *incremental build* button. This will produce a .DLM file, not the .MDL files. This is a small compiled file & is the last thing Matlab creates before it looks for the target. Of course it won't find the target, but it will create that file. This .DLM file will be the file you upload to the iLab.

SERVICE BROKER LOGON

Go to http://ilab.students.itee.uq.edu.au/ServiceBroker

You will be greeted with the logon screen where you should enter your username and password. This is the password that was emailed to you, not your normal ITEE password. (You can change it to your usual password using the "My Account" section once you login)



You will be brought to the “My Labs” page by default. Click on the “Launch Lab” button. It could be slow the first time you do this, however if nothing happens make sure you have installed Java.

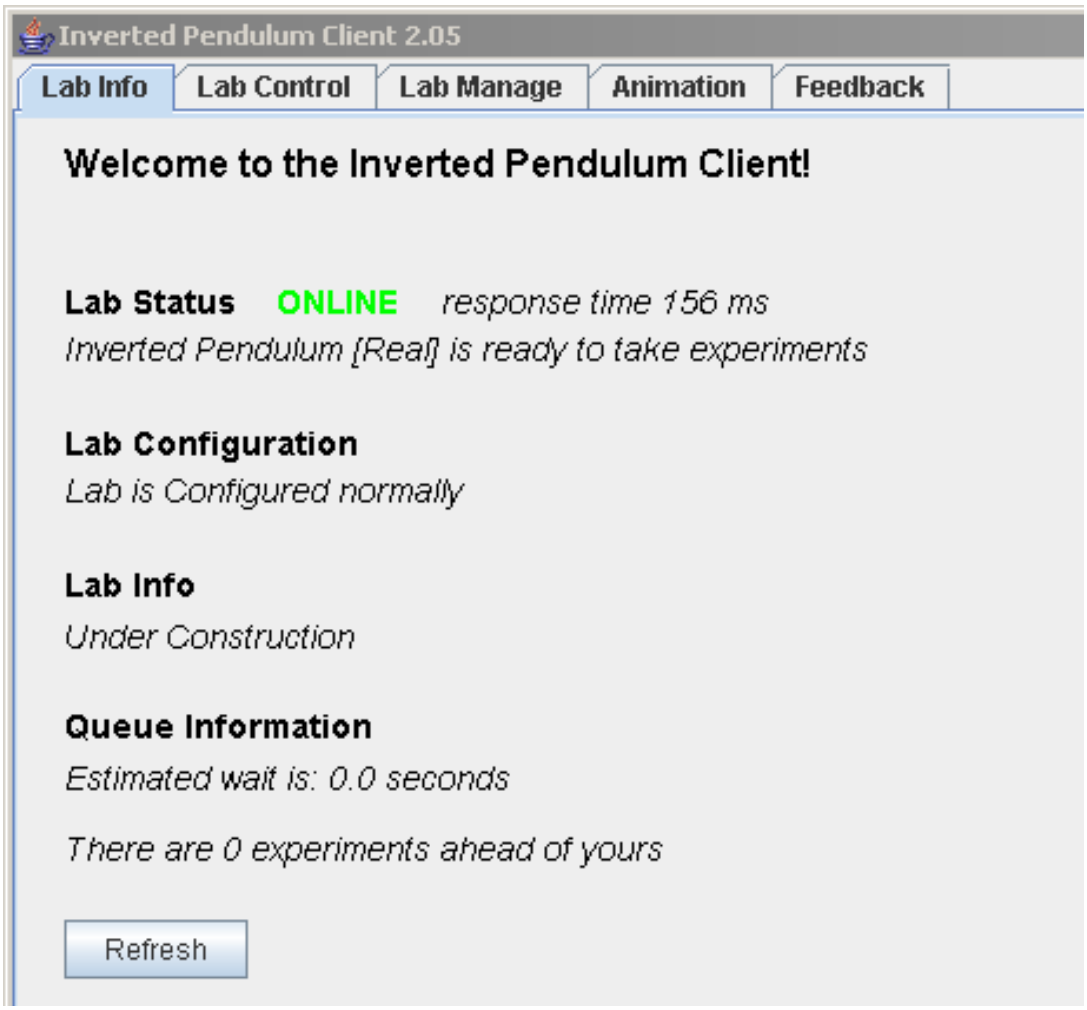
A popup will ask you if you trust this applet, click yes.

The applet will open, but the browser may still have focus so you may have to select the “Inverted Pendulum” applet from the task bar to bring it up.

A known issue is that occasionally it will open in a tiny box in the top left corner instead of the proper size. Click maximise and everything should be fine. If at any stage the client doesn’t look right or goes blank, just adjust the size of the client, minimise, maximise or drag the edges. Any change in the client window size causes it to repaint and reconfigure itself.

LAB CLIENT

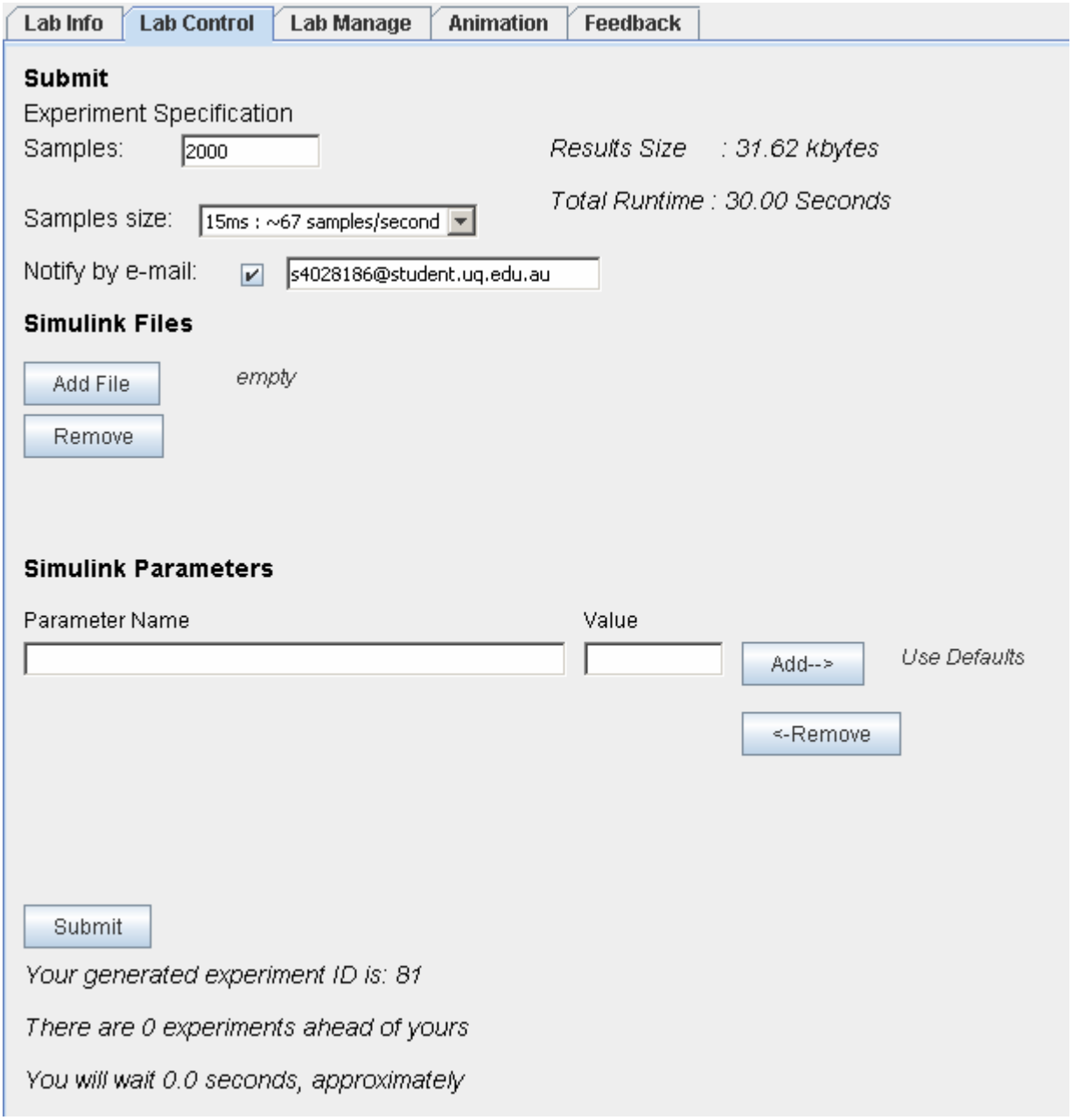
LAB INFO



The main points of interest are;

> *Lab Status* which confirms that the Lab Server is online & waiting for your experiment.
> *Queue Info*, how many experiments and how long before your experiment runs.

LAB CONTROL



Generally you need only enter how many samples of data you wish to take , the file to upload and your email address.

30 seconds worth of data is as much as you need. The experiment will run until it has taken the number of samples you specify or until it does into the failsafe state.

If you do not specify a file to upload it will run Dr. Hovland's solution.

Whilst it is not required, it is recommended that you click the "notify by email" option and enter an email address. A message will be delivered here when the experiment is complete. This email also contains information describing what happened during the course of the experiment. If there are any errors you will get there details in this email. For this reason it is recommended that you turn it on. It will remember your email next time you logon.

When you've done this you may click submit and should see a summary of your experiment & its experiment ID. You will need this ID # to play back the experiment when it is done.

Advanced options on the Lab Control pane

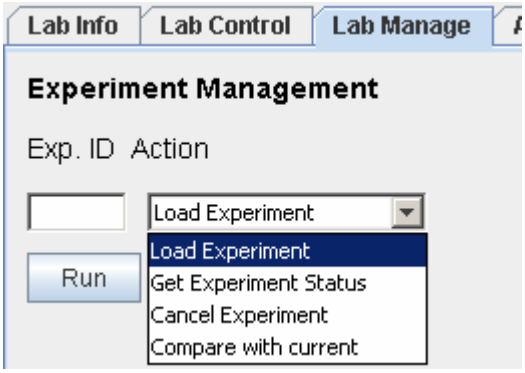
Simulink parameters
If for some reason you wish to alter parameters after you have compiled you may enter then in on the Java client.

For instance if you have a parameter called “pendulum/part1/signal1” that you wish to change to 0.2 then you would enter that into the name & value fields respectively and press add. It is forgiving regarding the direction of the slash and you may even use a full stop instead of a slash. If you wish to specify a 1-dimensional matrix you can do so by typing the values separated by commas.

Generally however you will not use this feature as you would already be familiar and more comfortable altering your parameters through simulink before compile time. However if you are tweaking and wish to change values rapidly without a re-compile this could be a good option.

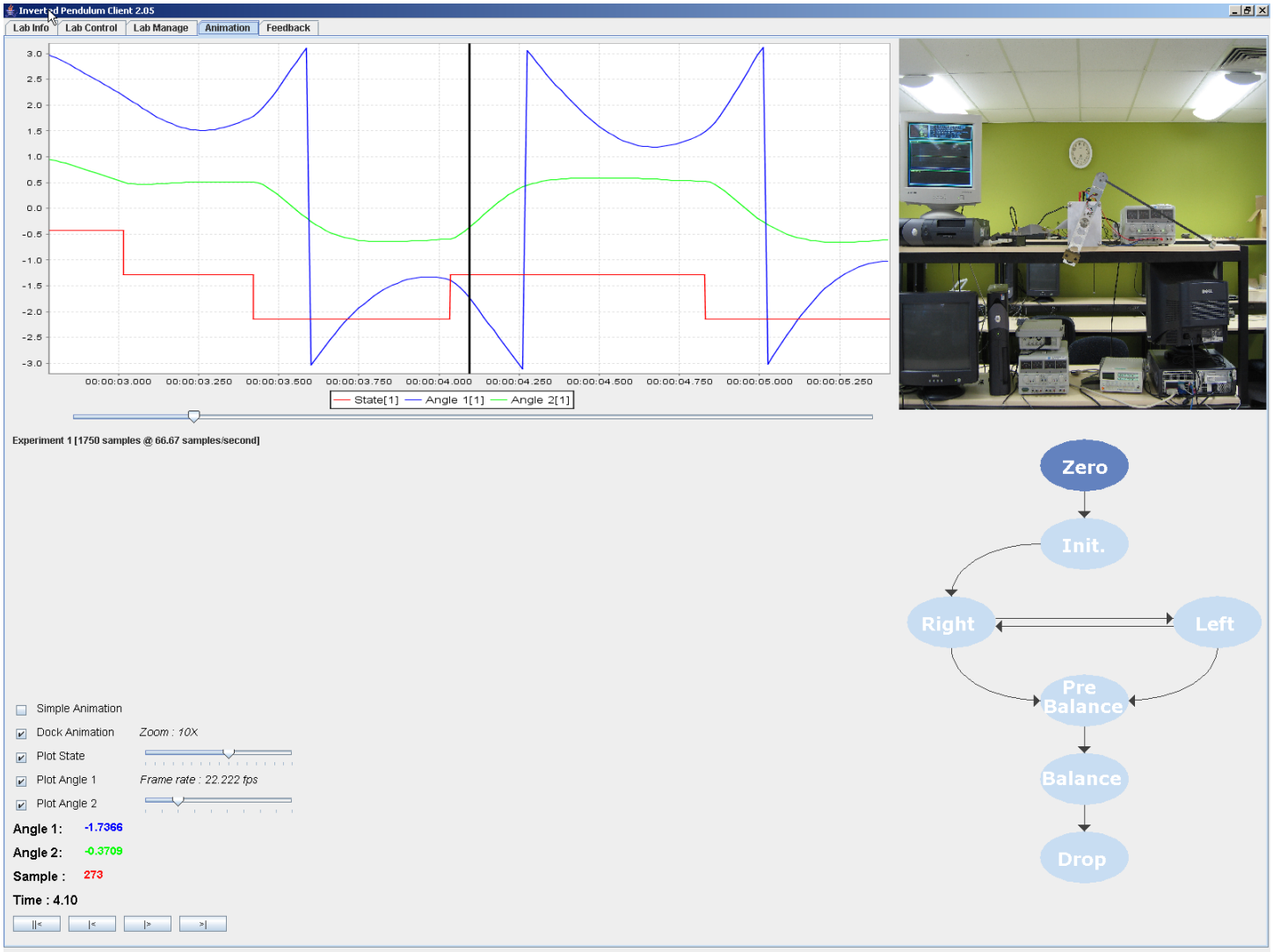
Sample Size
The default is 15ms samples, which results in a resolution of ~67 frames/second when you reanimate the results later. This is fine as you can alter the playback rate later according to the speed of your machine.

LAB MANAGE
This is the tab you use to play back results and view the status of your experiment through.



Load Experiment runs the experiment entered in the Exp. ID field only.
Get Experiment Status tells you whether the experiment has finished, is still in the queue etc.
Cancel Experiment removes your experiment from the queue
Compare with current if you already have one experiment loaded and wish to load another on top.

ANIMATION
Where the action happens. When you load an experiment the animation frame will pop up and the chart will appear with values plotted.



<u>*State Diagram :*</u> Animates the state assuming you have stuck to Dr Hovlands state diagram. The dark blue oval is your current state and the green arrow indicates how it got there.

<u>*Data Labels :*</u> These labels display the data in numeric form. Angle 1 (q2), Angle 2 (q1), the sample number and the current time. The colours of the angles also correspond to the colours on the pendulum animation in simple mode.

<u>*Time Marker :*</u> The black line down the centre of the chart indicates the current point in time. It will refresh every second whilst the animation is playing, or with each step or reset.

<u>*Control Options*:</u>

Simple Animation : If selected this is selected the pendulum is represented as two lines and a ball instead of using the photo animation.

Dock Animation : If selected the animation will occur docked within the Animation panel as shown in the diagram above. Otherwise it will have its own popup frame.

Plot x : Turns series' on and off on the chart

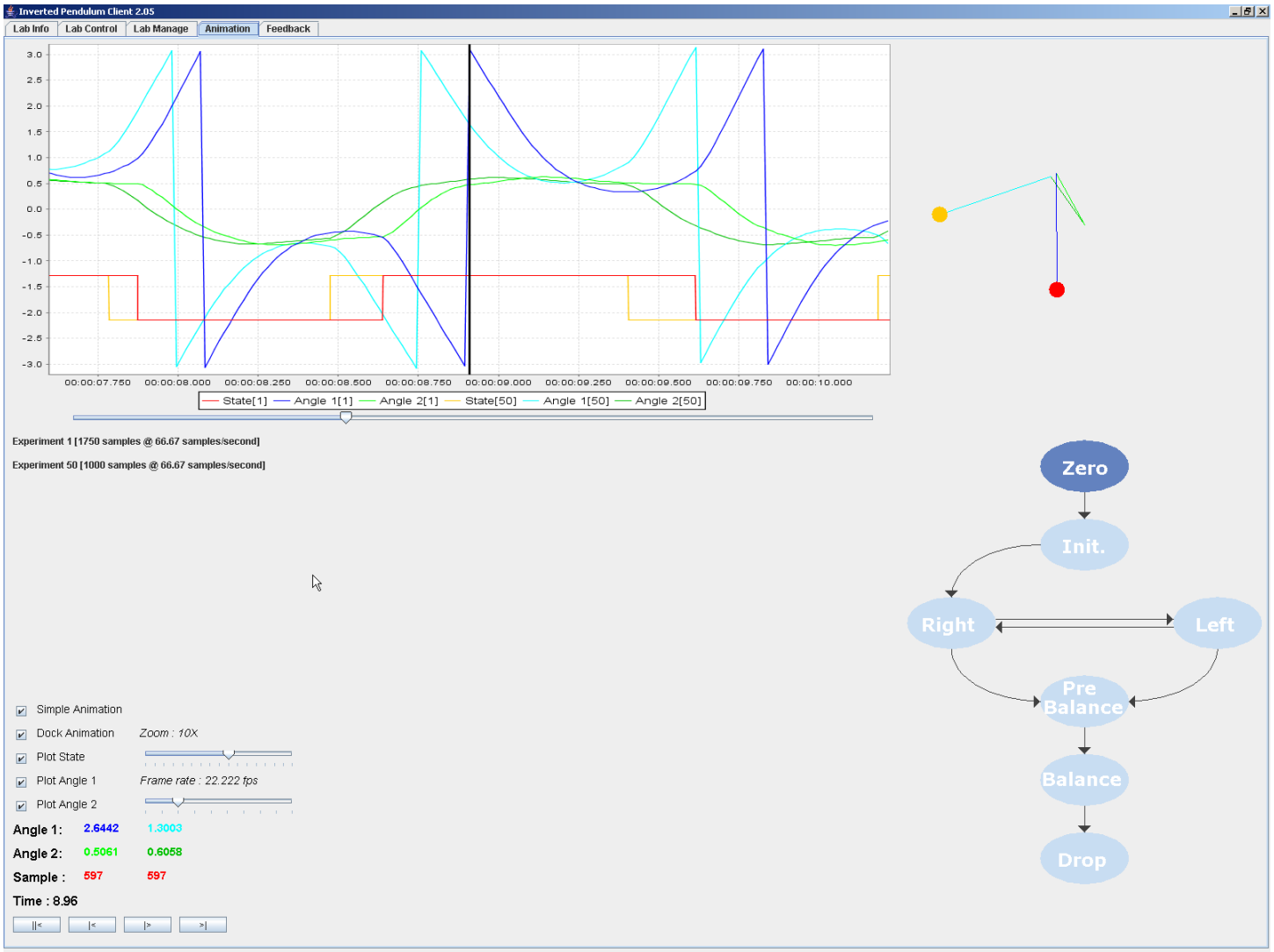
Zoom Slider : The zoom is set to 10x by default. It is recommended that you keep the zoom at that level or higher. The data is easier to see and it is easier on your CPU.

Frame rate Slider: Defaults to around 20 frames/second. There is generally no need to turn it higher and most computer will begin to lag if you do so. It shouldn't drop frames, but the animation will slow down.

Start, stop, step, reset buttons : Standard navigation buttons. Start/Stop is the same button. Step goes on frame (not sample) forward or back. If you wish to step through with a higher resolution turn the frame rate up. Reset takes it back to the start.

Time Slider : The far left is always time 0 and the far right is always the end. Moves the current time, and displays the angle, sample and time information, however the pendulum will not reanimate until you press play or step.

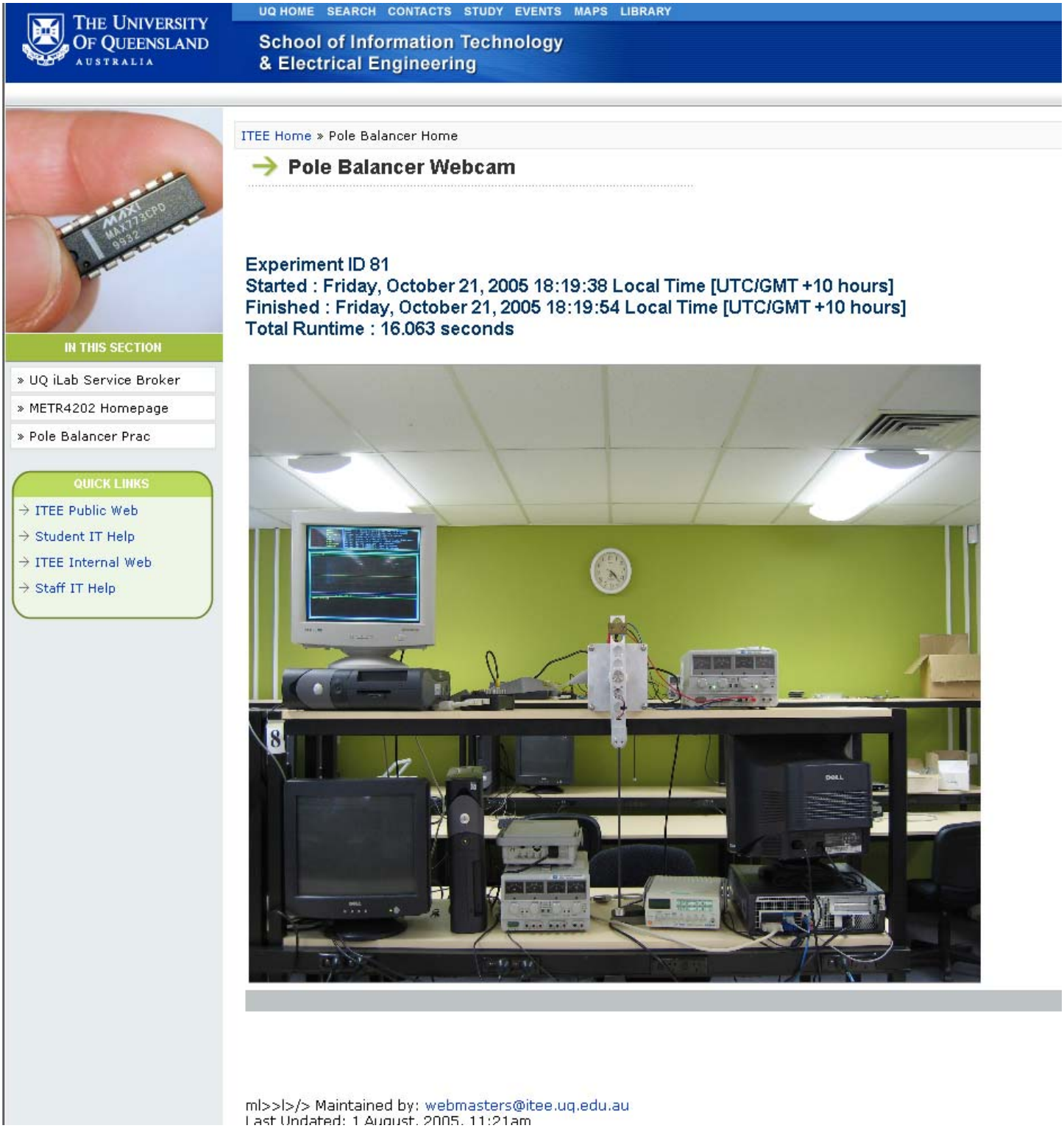
COMPARING EXPERIMENTS



When you use the *compare with current* option from the *Lab Manage* frame it will load both sets of data into the animation frame. Whilst no limit is set on how many you can load concurrently, the colours used will repeat after three datasets and your CPU will thank you to keep it under this number. The state diagram will only follow the first experiment you loaded, but the chart, pendulum animation and data displays will follow all experiments. It is recommended to use the *Simple Animation* for multiple experiments as the different colours make it easier to distinguish between the loaded datasets.

FEEDBACK
Possibly the most important frame of all? Tell us what you think in this section. The values will be loaded with what you entered last time you filled out the survey. If your feelings change, change the form and press “Update my response”

WEBCAM
http://130.102.73.13/LabServer/webcam.htm (Java applet & experiment information)
http://ilab-webcam.students.itee.uq.edu.au (Active X, Windows only, higher framerate, no exp. Info)


THE UNIVERSITY OF QUEENSLAND
AUSTRALIA
UQ HOME SEARCH CONTACTS STUDY EVENTS MAPS LIBRARY
School of Information Technology & Electrical Engineering
IN THIS SECTION
» UQ iLab Service Broker
» METR4202 Homepage
» Pole Balancer Prac
QUICK LINKS
ITEE Public Web
Student IT Help
ITEE Internal Web
Staff IT Help
ITEE Home » Pole Balancer Home
Pole Balancer Webcam
Experiment ID 81
Started : Friday, October 21, 2005 18:19:38 Local Time [UTC/GMT +10 hours]
Finished : Friday, October 21, 2005 18:19:54 Local Time [UTC/GMT +10 hours]
Total Runtime : 16.063 seconds
ml>>l>/> Maintained by: webmasters@itee.uq.edu.au
Last Updated: 1 August, 2005, 11:21am

This display what is currently going on and what experiment is running or just finished running. This is made largely obsolete by the animation, but is a useful eye to have if there are physical problems with the pendulum.